VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Systém pro statistické vyhodnocování parametrů 3D povrchů

Statistical Evaluation of Three Dimensional Surfaces

2010 Petr Baroš

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

....

Datum Petr Baroš v.r.

Tímto bych rád poděkoval vedoucímu své bakalářské práce, Ing. Michalovi Krumniklovi, za odborné vedení a podnětné rady, které mi v průběhu práce uděloval.

Abstrakt
Cílem bakalářské práce bylo vytvořit informační systém pro správu měřených 3D povrchů a statistické vyhodnocování jejich parametrů, který by byl dostupný pomocí internetových technologií. Text popisuje frameworky a programy použité k tvorbě informačního systému a také knihovny zobrazující 3D model uložených digitalizovaných povrchů. Čtenář je krok za krokem seznámen s budováním klient/server aplikace, databázových struktur a komunikačních komponent za použití v dnešní době moderních programovacích technik a technologií. V závěrečné kapitole je stručně popsána instalace a konfigurace celého systému.

Klíčová slova
Informační systém, internetové technologie, internetová aplikace, webové služby, počítačová grafika, 3D zobrazení, statistika, Microsoft, Silverlight, SQL server, IIS

Abstract
The goal of this thesis was to create information system for management of scanned 3D surfaces a statistical evaluation of their parameters. The system should be accessible from WWW. Text describes frameworks and programs used to develop information system and libraries displaying 3D model of digitalized surfaces. Step by step reader is acquainted with implementation of client/server application, database structures and communication components using today's modern programming techniques and technologies. The installation and configuration of the system is explained in the last chapter.

Keywords
Information system, internet technology, internet application, web service, computer graphics, 3D projection, statistics, Microsoft, Silverlight, SQL server, IIS

Seznam použitých symbolů a zkratek

CLR – Common Language Runtime – virtuální stroj v .NET Framework

DNS – Domain Name System – systém doménových jmen

FTP, FTPS – File Transfer Protocol (Secure) – protokol pro přenos souborů

HTTP, HTTPS – Hypertext Transfer Protocol (Secure) – protokol pro přenos dokumentů

IIS – Internet Information Service – služba spravující webové servery

LINQ to SQL – objektově relační mapování v .NET Framework 3.5

Moonlight – projekt Silverlight pro operační systémy rodiny linux

MSIL – Microsoft Intermediate Language – intermediární jazyk .NET Framework

.NET Framework 3.5 – obsahuje základní knihovní třídy a metody.

NNTP – Network News Transfer Protocol – protokol pro přenos diskusních skupin

ORM – Object Relation Mapping – objektově relační mapování

Sa [m] – průměrná nerovnost povrchu

Sdq [°] – vyjadřuje střední kvadratickou odchylku zvrásnění povrchu

Sdr [%] – procento přidané plochy, díky struktuře povrchu, k ideální ploše snímaného povrchu

Sds [n/m] – hustota špiček, čili počet špiček na jednotku plochy povrchu

Sq [m] – kvadratická nerovnost povrchu

Sku – míra fluktuace kolem střední hodnoty struktury povrchu

Ssc [m] – průměr zakřivení špiček povrchu

Ssk – distorze struktury povrchu

Sz [m] – průměrná odchylka mezi 5 nejvyššími špičkami a 5 nejhlubšími prohlubněmi povrchu

Silverlight 3 – framework, obsahující omezené knihovní třídy a metody z .NET Framework 3.5, pro použití v internetových aplikacích.

SMTP – Simple Mail Transfer Protocol – protokol pro přenos zpráv

SQL server – systém řízení báze dat firmy Microsoft

TCP – Transmission Control Protocol – přenosový protokol transportní vrstvy

Telnet – Telecommunication Network - protokol pro spojení klient/server pomocí TCP

URL – Uniform Resource Locator – řetězec, specifikující umístění zdrojů informací

WCF – Windows Communication Foundation –knihovna pro přenos dat, součást .NET Framework 3.5

WPF – Windows Presentation Foundation – knihovna uživatelského rozhraní, součást .NET Framework 3.5

WSDL – Web Service Description Language – jazyk popisující funkce webové služby, její datové struktury a komunikační protokoly

XAML – Extensible Application Markup Language – deklarativní jazyk založený na XML, popisující grafické uživatelské rozhraní.

XAP – soubor Silverlight aplikace

Obsah

1 Úvod

Naším úkolem je vytvořit informační systém, který bude spravovat měřené vzorky povrchů. Systém bude počítat statistické parametry, které následně zobrazí společně s nasnímaným obrázkem povrchu, či jeho 3D reprezentací. Můžeme si uvést několik příkladů, jak ho lze využít v praxi. Komerční – firma, zabývající se výrobou textilií, při závěrečné kontrole kvality snímkuje povrch textilie, tyto vzorky odesílá do našeho systému, ten je analyzuje a vyhodnocuje jejich kvalitu. Vědecký – tým, zabývající se vývojem nových materiálů, eviduje snímky jeho povrchů, materiál např. zahřejí na vysokou teplotu, či naopak ochladí na nízkou a nový snímek povrchu uloží do našeho systému, následně sledují změny povrchu materiálu. Lékařský – při sledování účinnosti léčby kožního onemocnění si lékař pořizuje snímky, ty ukládá do našeho systému, který počítá statistický parametry rozsahu a intenzity onemocnění, porovnáním několika vzorků je lékař schopen určit, jak je léčba účinná.

V následující kapitole popíšeme statistické parametry, které lze u povrchu počítat. Řekneme si, k čemu jsou dobré, jak nám mohou pomoci analyzovat povrch a také jejich matematickou interpretaci. Dále se seznámíme s několika informačními technologiemi, které nám pomohou vybudovat náš systém. Prvním z nich je .NET Framework 3.5, který je základním stavebním systémem celého systému. Jeho součástí je Windows Communication Foundation (WCF), na kterém postavíme veřejné rozhraní webové služby – serverové části aplikace, LINQ to SQL implementující objektově-relační mapování databáze a v neposlední řadě i Windows Presentation Foundation (WPF), sloužící jako základ prezentační vrstvy. Přiblížíme si internetovou technologii Silverlight, která vytvoří klientskou část aplikace. Nesmíme opomenout produkty SQL Server 2005 a Internet Information Services (IIS), jako i další podpůrné knihovny.

V další části textu postupně vytvoříme klient/server aplikaci podle předem připravené analýzy a designu. Prakticky použijeme znalostí, získané v předešlém textu a pomocí vývojových nástrojů implementujeme informační systém. V neposlední řadě se seznámíme s instalací a konfigurací jednotlivých součástí.

2 Teoretický základ

Tato kapitola nás seznámí s teoretickými základy, kterých bylo použito při tvorbě informačního systému. Vysvětlíme si jednotlivé statistické parametry povrchu, jejich význam i samotný výpočet. Popíšeme si rovněž standardní SW a knihovny, na kterých je systém postaven. A nakonec si přiblížíme knihovny třetích stran, jež se v systému využívá k vytvoření specifických vlastností a funkcí.

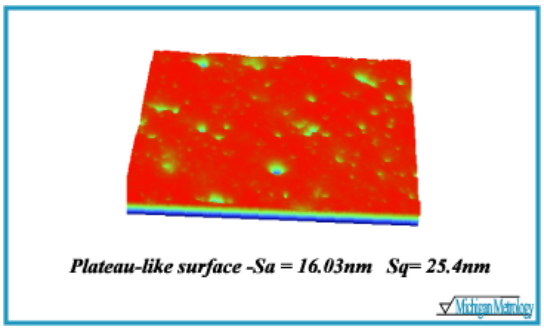
2.1 Statistické parametry povrchu

Jelikož se nepovažuji za odborníka na statistické parametry 3D povrchů, tak v této kapitole budu pouze citovat dostupnou literaturu [4].

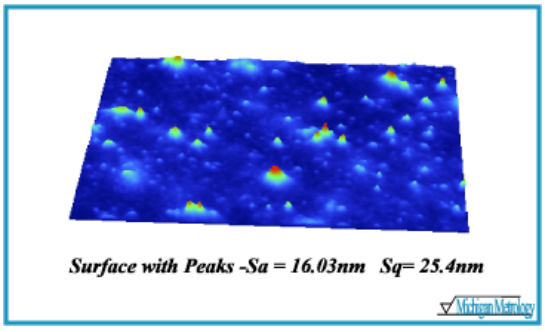
Sa a Sq

Vyjadřují průměrnou a střední kvadratickou nerovnost (drsnost) povrchu. Jsou vyhodnoceny ze všech jeho 3D bodů. Matematicky se počítají jako:

$$Sa = \iint_a |Z(x,y)|dxdy$$



$$Sq = \iint_a \sqrt{(Z(x,y))^2}dxdy$$



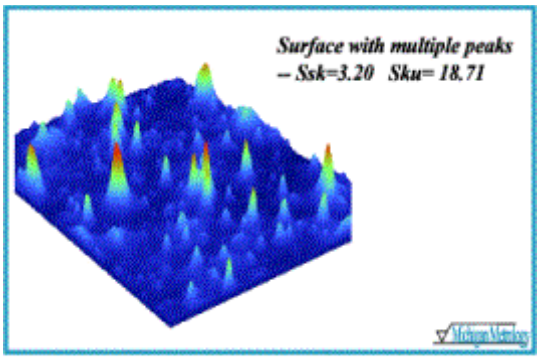
Použití – Sa a Sq parametry vyjadřují celkovou míru zrnitosti povrchu. Parametry nejsou citlivé na různé špičky, prohlubně a další zrnitosti. Předchozí obrázky zobrazují dva velmi rozdílné povrchy, přesto hodnoty Sa a Sq jsou pro oba stejné. Nicméně pokud je typ povrchu již určen,

mohou tyto parametry indikovat významné odchylky ve struktuře povrchu. Sq je typicky používáno pro optické povrchy a Sa pro mechanické.

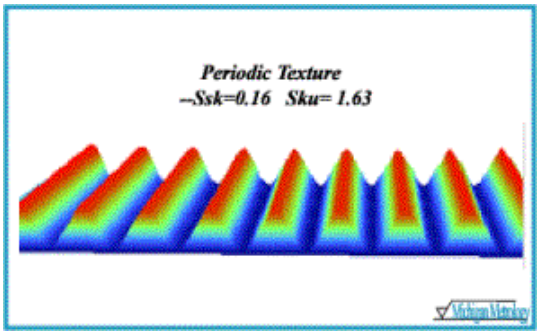
Ssk a Sku

Ssk a Sku jsou distorze a míra fluktuace kolem střední hodnoty struktury 3D povrchu. Přeneseně lze říct, že pokud je histogram výšek všech měřených bodů symetrický (např. Gausova křivka), pak Ssk a Sku reprezentují odchylky od této ideálně symetrické křivky. Matematicky se počítají jako:

$$Ssk = \frac{1}{Sq^3} \iint_a \left(Z(x,y)\right)^3 dxdy$$



$$Sku = \frac{1}{Sq^4} \iint_a \left(Z(x,y)\right)^4 dxdy$$

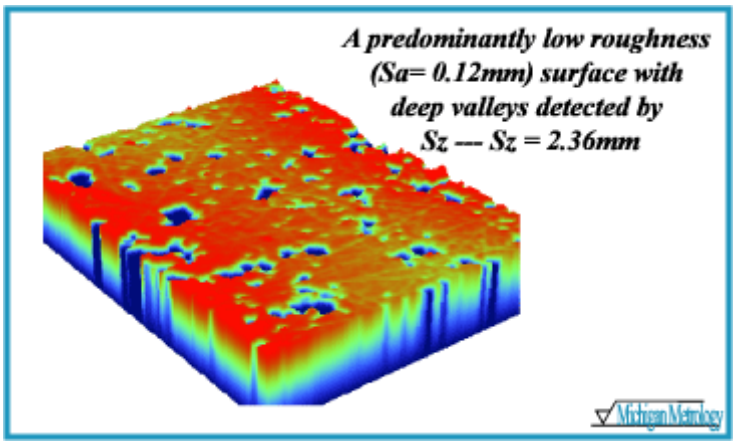


Použití – Ssk reprezentuje stupeň symetrie měřených výšek povrchu kolem střední roviny. Kladná hodnota (Ssk > 0) vyjadřuje převahu špiček na povrchu, naopak záporná hodnota (Ssk < 0) vyjadřuje převahu prohlubní. Pokud je Sku > 3.0 pak vyjadřuje přítomnost nadměrně vysokých špiček, či nadměrně hlubokých prohlubní v povrchu, v opačném případě (Sku < 3.0) jejich nedostatek. Pokud jsou výšky povrchu symetricky rozloženy, pak Ssk = 0.0 a Sku = 3.0. Povrchy charakteristické pozvolnou změnou výšky, bez přítomnosti extrémních špiček, či prohlubní mívají Sku < 3.0. Ssk je užitečné k vyhodnocení, jak je povrch vybroušený, a ke sledování míry opotřebení. Sku je užitečné k indikaci přítomnosti vad, jako jsou špičky, či prohlubně, přítomné na povrchu.

Sz

Sz je průměrná odchylka mezi 5 nejvyššími špičkami a 5 nejhlubšími prohlubněmi celého 3D povrchu. Špička je definována jako bod, který je vyšší než všech 8 jeho okolních bodů. Prohlubeň je definována jako bod, který je nižší než všech 8 jeho okolních bodů. Matematicky se počítá jako:

$$Sz = \sum_{1}^{5} |výška\ špiček| + \sum_{1}^{5} |hloubka\ prohlubní|$$



Použití – Sz je užitečné k popisu „obalové plochy“, která obsahuje většinu měřených výšek povrchu, obzvlášť pokud Sa a Sq dominují díky obecné zrnitosti povrchu. Sz může indikovat změnu dříve než Sa, nebo Sq, pokud je povrch měněn, jako např. při studiu opotřebení.

Sds

Sds, neboli hustota špiček, je počet špiček na jednotku plochy povrchu. Matematicky se počítá jako:

$$Sds = \frac{počet\ špiček}{plocha}$$

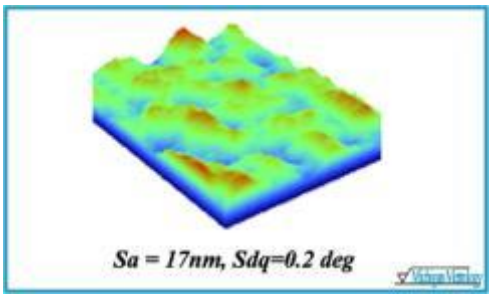
Použití – Sds je klíčový parametr pro zjišťování přilnavosti a kvality elektronických kontaktů. Způsob, jak se špičky elasticky a plasticky deformují při zatížení, určuje právě Sds parametr. Povrchy s malým Sds jsou při vysokého namáhání více náchylné k důlkové korozi, či ke vzniku

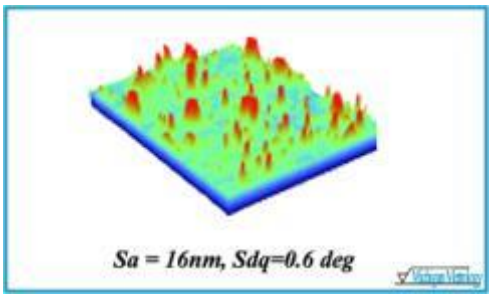
úlomků a odštěpků. Hustota špiček může rovněž ovlivňovat kvalitu vzhledu, pokud je povrch natřen barvou.

Sdq a Ssc

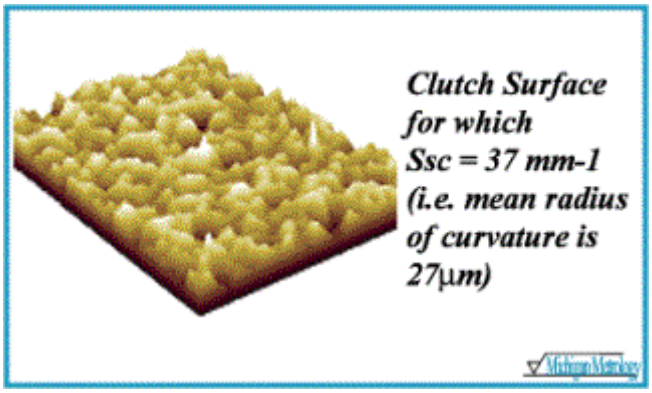
Sdq vyjadřuje střední kvadratickou odchylku zvrásnění povrchu. Ssc je průměr zakřivení špiček povrchu uplatněných pro Sds kalkulaci. Matematicky se počítají jako:

$$Sdq = \sqrt{\frac{1}{A}\int_0^{Lx}\int_0^{Ly}\left(\frac{\partial Z(x,y)}{\partial x}\right)^2 + \left(\frac{\partial Z(x,y)}{\partial y}\right)^2 dxdy}$$





$$Ssc = \frac{1}{N}\iint\left(\frac{\partial^2 Z(x,y)}{\partial x^2}\right) + \left(\frac{\partial^2 Z(x,y)}{\partial y^2}\right) dxdy$$

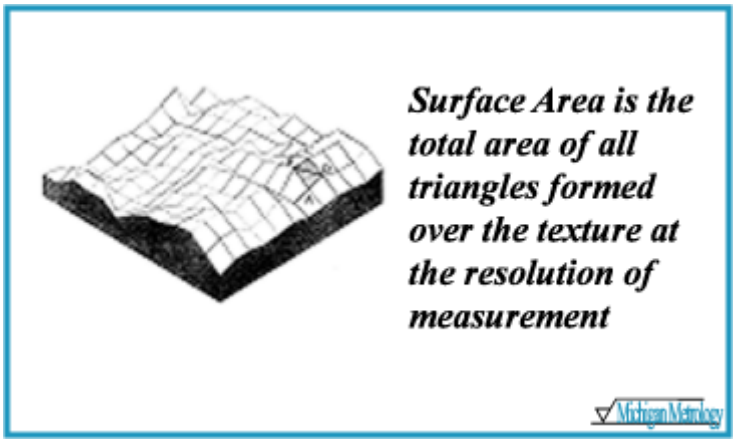


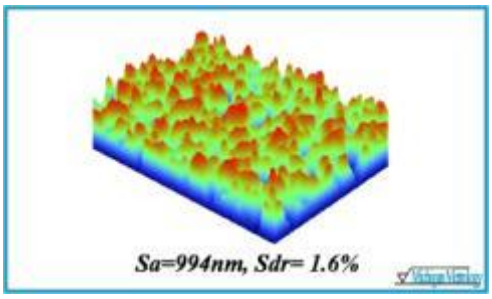
Použití – Sdq celková míra zvrásnění povrchu a může být použita k rozlišení povrchu s podobnou průměrnou nerovností Sa, jak je ukázáno na předešlých obrázcích. Sdq může najít uplatnění při posuzování kosmetického vzhledu. Ssc je užitečná v předvídání elastické a plastické deformace povrchu při různém zatížení a může tak definovat předpokládané opotřebení materiálu.

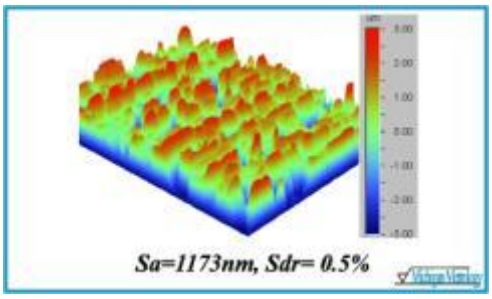
Sdr

Sdr vyjadřuje procento přidané plochy, díky struktuře povrchu, k ideální ploše snímaného povrchu.

$$Sdr = \frac{(plocha\ povrchu) - (digitalizovaná\ plocha\ povrchu)}{digitalizovaná\ plocha\ povrchu}$$







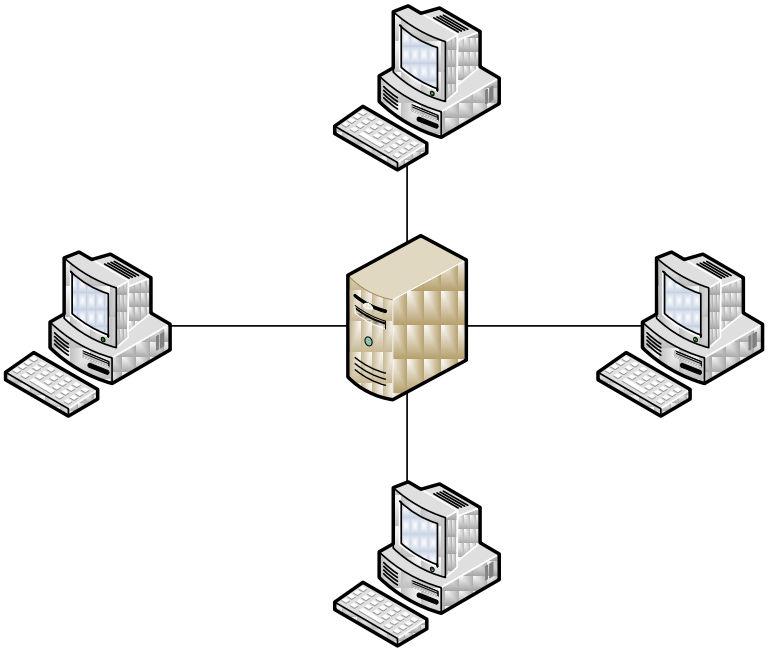
Uplatnění – Sdr může rozlišit povrchy s podobným rozkmitem a průměrem nerovnosti. Sdr typicky roste s prostorovou složitostí struktury povrchu, přestože hodnoty Sa jsou podobné. Sdr je užitečný při hledání povrchů s vysokou přilnavostí, rovněž může být významný při určování chování materiálu ve styku s mazadly a jinými kapalinami.

2.2 Aplikace typu klient/server

Klient/server je síťová architektura, která se skládá z

- Klienta – program, často s uživatelským rozhraním, ale může se jednat i o tzv. robota, čili program pracující samostatně bez uživatelského rozhraní
- Serveru – program, většinou bez uživatelského rozhraní

Tyto programy spolu komunikují přes počítačovou síť. Model klient/server používá většina obchodních či firemních aplikací, či databází, můžeme také uvést internetové protokoly HTTP, SMTP, Telnet, DNS, apod.



Obrázek 2-1: Klient/server architektura

Klient žádá o služby server, který mu následně, po zpracování požadavku, odpoví a vrátí výsledek. Tak například webový prohlížeč je klient, který žádá webový server o zpřístupnění informací. Pokud chceme vědět, jaké je právě počasí v Kalifornii, kam se zrovna chystáme na služební cestu, v prohlížeči zadáme URL serveru, poskytující informace o počasí na americkém kontinentu, prohlížeč pak následně komunikuje se serverem a získané informace zobrazí. Tato informace je přístupná kterémukoli klientovi, webovému prohlížeči, na celém světě a informace o počasí v Kalifornii je zobrazována všem uživatelům stejně.

Klient

- Aktivní přístup – zasílá žádosti na server
- Čeká na odpovědi ze serveru a ty pak poskytuje uživatelům
- Komunikuje s uživateli, pomocí uživatelského rozhraní

Server

- Pasivní – naslouchá na síti a reaguje na žádosti připojených klientů
- Po přijetí dotazu, jej zpracuje a odešle odpověď

K jednomu serveru, může přistupovat (ne)omezené množství klientů. A jeden klient může žádat o služby (ne)omezené množství serverů. Vše záleží na konfiguraci a na zdrojích, které má program k dispozici.

Výhody

- *snadnější údržba* – architektura klient/server rozdělí činnosti a zodpovědnosti počítačového systému mezi několik programů, které spolu komunikují prostřednictvím sítě. Je jednoduché opravit činnost serveru (pokud zůstane zachováno rozhraní), bez nutnosti instalovat nové klienty. A naopak při úpravě klienta (který je většinou jednoduchý) není třeba zasahovat do složitých systémů na serveru.
- *bezpečnost* – data jsou uložena na serverech, jež jsou mnohem bezpečnější. Servery pak kontrolují přístup k informacím např. pomocí autorizace uživatele.
- data a informace jsou uložena na jednom místě a všichni klienti je získávají z jediného centralizovaného zdroje.

Nevýhody

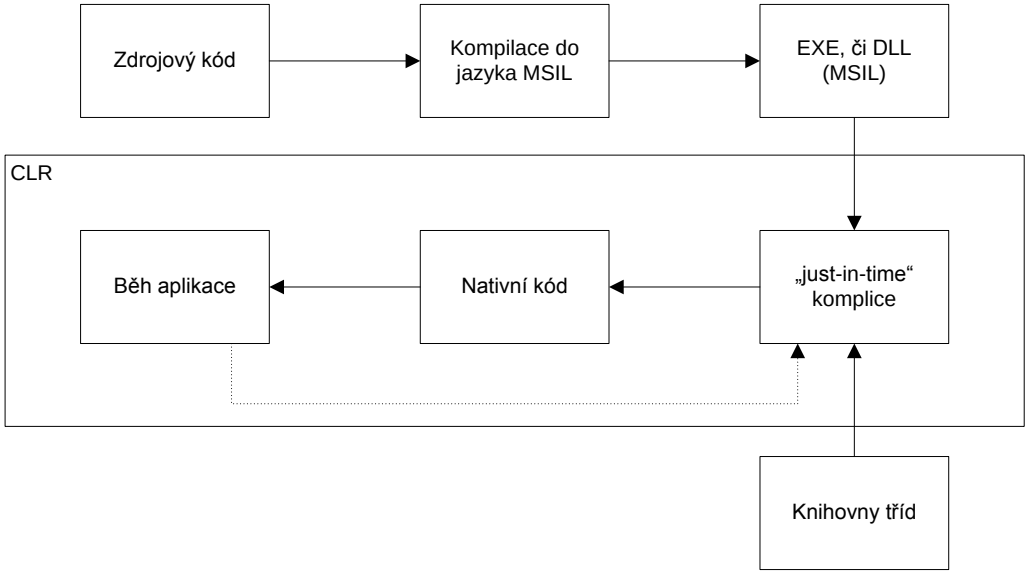
- *výkon* – pokud je k serveru připojeno velké množství klientů a server má potíže vyřizovat jejich požadavky, stane se „úzkým hrdlem“ systému a může způsobit dokonce kolaps systému. To lze částečně vyřešit zvýšením přiřazených zdrojů, či rozdělením činností mezi více serverů.
- *dostupnost* – pokud dojde k výpadku serveru, dotazy klientů nemohou být vykonávány a opět to způsobí nefunkčnost systému. To lze vyřešit vytvořením sekundárních (neboli zrcadlových) serverů, na kterých běží identický program a na který se požadavky přesměrují v případě výpadku serveru primárního.

2.3 Použitý software a technologie

2.3.1 .NET Framework 3.5

Microsoft .NET Framework může být instalovaný na operačních systémech platformy Windows (včetně verzí pro mobilní zařízení). Obsahuje velkou knihovnu kódu, řešící běžné programovací problémy a rovněž virtuální stroj, vykonávající programy napsané pro tento framework.

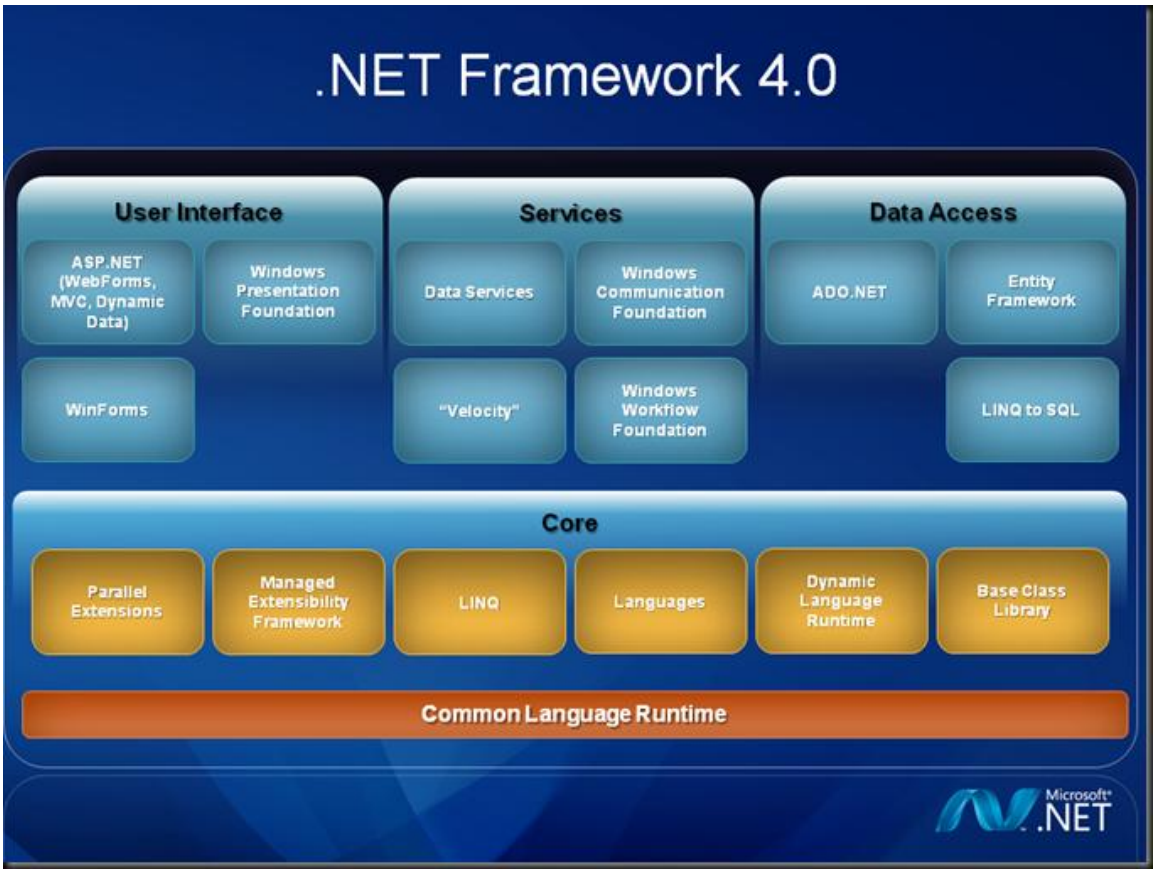
Součástí .NET Frameworku je Common Language Runtime (CLR), který spouští a řídí kód programu a vykonává požadavky programu [1]. Programy nejsou kompilovány přímo do nativního kódu, nýbrž do intermediárního jazyka (MSIL). Názorné vysvětlení celého procesu popisuje Obrázek 2-2.



Obrázek 2-2: CLR – zdroj [1]

Základní knihovna tříd poskytuje velké množství prvků jako uživatelské rozhraní, přístup k datům a databázím, kryptografické techniky, podporu pro vývoj webových aplikací, matematické algoritmy, třídy pro práci se sítí a další. Knihovny .NET Frameworku spolu s vlastním kódem umožňují programátorům vytvářet vlastní aplikace. Většinu informací o .NET Framework lze najít na internetových stránkách .NET komunity [8].

V průběhu let byly postupně vydány verze 1.0, 1.1, 2.0, 3.0, 3.5 a v čase psaní textu byla vydána zatím poslední verze 4.0. Tato verze přináší hlavně nové třídy pro paralelní programování, vykonávání lambda výrazů, práci s velkými celými čísly a komplexními čísly a další. Diagram základních prvků .NET Frameworku 4.0 zobrazuje Obrázek 2-3.



Obrázek 2-3: Architektura .NET Framework – zdroj [5]

Mezi nejpoužívanější programovací jazyky .NET Frameworku patří:

- C#
- Visual Basic
- Delphi (pascal – není produkt firmy Microsoft)

Méně běžné programovací jazyky, které jsou součástí MS Visual Studia pro .NET:

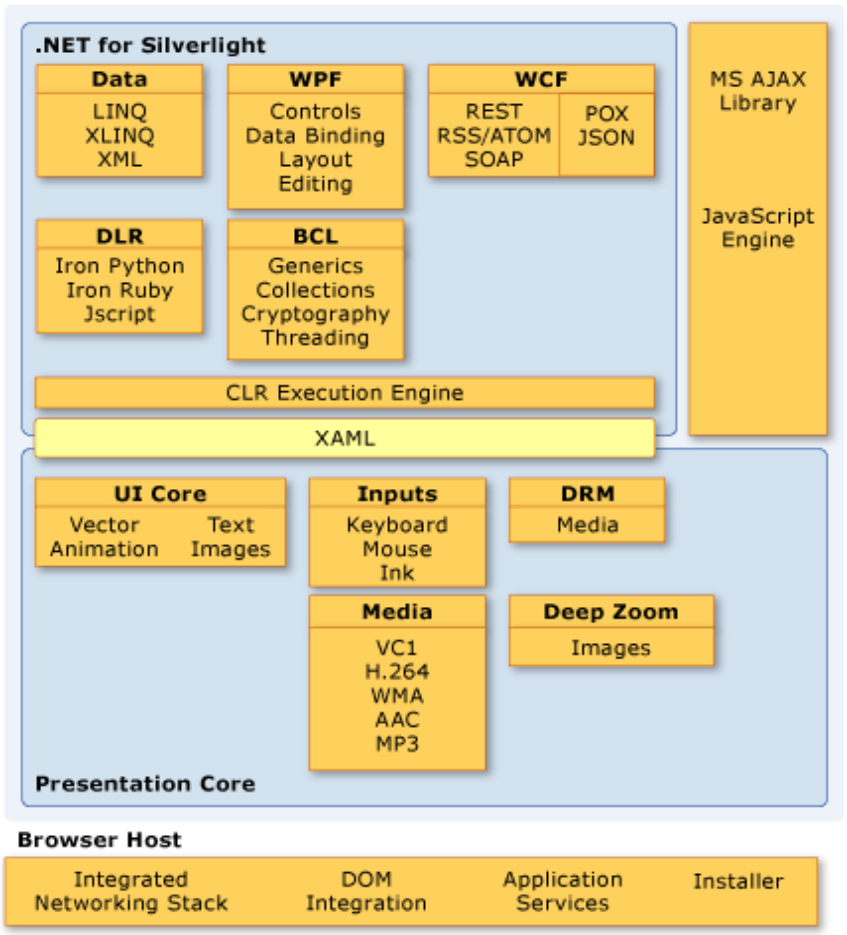
- C++
- F#
- J#
- IronPhyton, IronRuby

2.3.2 Silverlight 3

Na oficiálních webových stránkách Microsoftu píší, že Silverlight je implementace .NET Frameworku, která je „cross-browser“, neboli nezávislá na webovém prohlížeči, a „cross-platform“, neboli nezávislá na operačním systému. Obě tvrzení mají však jistá omezení, v současné době jsou dostupné verze pro operační systémy Windows a MacOS a pro webové

prohlížeče Internet Explorer, Firefox a Safari. Pro operační systémy linux existuje projekt Moonlight, který zajišťuje kompatibilitu se Silverlight.

Většina tříd .NET Frameworku je implementována v Silverlight a to i přes bezpečnostní omezení (nelze přistupovat k lokálním zdrojům, nelze spouštět externí programy a další), kterým se internetová technologie nevyhne. Pro programátory to tedy obecně znamená, že schopnosti a znalosti, které získali při vytváření desktopových .NET aplikací, mohou plně uplatnit při vývoji Silverlight internetové aplikace [3].



Obrázek 2-4: Architektura Silverlight 3 – zdroj [5]

Silverlight je přídavný modul, který se instaluje do webového prohlížeče (instalace .NET Frameworku na klientském počítači není vyžadována). Je postaven na technologii WPF a uživatelské rozhraní je vytvářeno pomocí deklarativního programovacího jazyka XAML, který kromě vzhledu uživatelského rozhraní může popisovat například i animace a vazbu na data.

2.3.3 SQL Server 2005 Express Edition a LINQ to SQL

SQL Server je relační databázový server vydaný firmou Microsoft. Hlavním dotazovacím jazykem jsou T-SQL a ANSI SQL.

.NET Framework 3.5 přinesl knihovní třídy pro relačně-objektové mapování pro databáze pracující s dotazovacím jazykem SQL, neboli LINQ to SQL. Klíčové pro pochopení jazyka LINQ je zejména znalost těchto programovacích konstrukcí:

- Lambda výrazy – jsou jednodušší metodou zápisu anonymních metod.
- Inicializátory objektů a kolekcí.
- Rozšiřující metody tříd.
- Anonymní třídy umožňující např. rychlé vytvoření objektů přenášejících informace vyžádané z databáze přes LINQ.
- Klíčové slovo `var`, nutná podmínka pro využití anonymních tříd.
- „Expression trees“ výrazové stromy umožňující za jistých podmínek kompilátoru vytvoření jeho objektové reprezentace místo vyhodnocení výrazu.

Jazyk LINQ lze obecně použít i pro kolekce, případně dotazování do xml souboru. Pro lepší pochopení uvedeme následující příklad v jazyce C#:

```
string[] jmena = {"Petr", "Pavel", "Jan", "Michal", "Martin"};
var kratkaJmena = from j in jmena
                  where j.Length < 5
                  orderby j.Length
                  select j;
foreach (string jmeno in kratkaJmena)
{
   Console.WriteLine(jmeno); // vypise „Jan“ a „Petr“
}
```

Použitím Lambda výrazů vypadá kód následovně:

```
string[] jmena = {"Petr", "Pavel", "Jan", "Michal", "Martin"};
var kratkaJmena = jmena.Where(j => j.Length < 5).OrderBy(j => j.Length);
foreach (string jmeno in kratkaJmena)
{
   Console.WriteLine(jmeno); // vypise „Jan“ a „Petr“
}
```

2.3.4 IIS - Internet Information Services

Internet Information Services (IIS) – původně pojmenovaný Internet Information Server – je webový server vytvořený firmou Microsoft instalovaný na operačních systémech Windows. Je to druhý nejpoužívanější webový server za Apache HTTP Server. Podporované protokoly: FTP, FTPS, SMTP, NNTP, a HTTP/HTTPS. Mimo jiné obsahuje:

- *Moduly HTTP* – provádí specifické úlohy pro HTTP, jako jsou odpovědi na dotazy klientů uložených v hlavičkách, návrat chybových hlášení, či přesměrování dotazů.

- *Moduly zabezpečení* – provádí úlohy odpovědné za bezpečnost, jako je specifikace autentifikačních schémat, URL autorizace a filtrování dotazů.
- *Moduly obsahu* – provádí úlohy jako návrat defaultní stránky, na blíže nespecifikovaný dotaz, odesílání statických souborů a výpis adresářů.
- *Moduly komprese* – provádí úlohy komprese a dekomprese přenášených dat s použitím Gzip metody.
- *Moduly cache* – provádí úlohy jako ukládání často používaných dotazů přímo v paměti.
- *Moduly logování a diagnostiky* – provádí úlohy jako zaznamenávání odeslaných informací a vytváření HTTP.sys logovacího souboru, pro výpis událostí a právě probíhajících dotazů.

2.3.5 Další knihovny

Následuje popis 3 knihoven externích tvůrců, kterých bylo použito při tvorbě informačního systému.

Kit3D

Mark Dawson vytvořil knihovní třídy pro Silverlight, které vytvářejí podobnou funkci jako třída Viewport3D a další v plnohodnotném .NET Frameworku 3.5. Tyto třídy umožňují definovat 3D model pomocí bodů, ploch a normál, přiřazovat plochám materiály, definovat polohu kamery a směr pohledu. Následně umí definované modely zobrazovat ve webovém prohlížeči. Přestože tento projekt zůstal nedokončený (poslední vydání na kit3d.codeplex.com je 20. 11. 2008, chybí například podpora osvětlování, apod.), byl pro použití v projektu dostatečný a povrchy lze s doplněnými funkcemi pro stínování zobrazovat ve 3D. A na povrch lze tak nahlížet z různých stran a získat lepší představu o jeho struktuře.

- Volná licence – „New BSD License (BSD)“

Log4net

Firma Apache vydala tuto knihovnu pro podporu logování v .NET programech. Jednoduchým způsobem, pomocí konfigurace, lze tak zapisovat do textového souboru informace o právě probíhajících událostech a procesech na serveru. Je zde rovněž implementována podpora vláken.

- Volná licence – GPL

ICSharpCode.SharpZipLib

Projekt ICSharpCode, který vytvořil knihovnu pro podporu Zip, GZip, Tar a BZip2 komprese a dekomprese souborů [2].

- Volná licence – GPL

3 Implementace systému

Systém spravuje měřené povrchy a jejich vypočítané statistické parametry. Jednotlivé povrchy jsou hierarchicky řazeny podle jména (lze si též představit jako složky – adresáře v souborovém systému). Každý povrch může mít libovolné množství vzorků (lze si též představit jako soubory v souborovém systému), které jsou identifikovány podle času uložení do systému. Tyto vzorky vyjadřují jedno měření (snímání) povrchu a každý se skládá z

- binárního souboru – digitalizovaného snímku povrchu (*.jpg,*. png, apod.)
- textového souboru – seznam 3D bodů rozpoznaných z digitalizovaného snímku
- vypočítaných statistických parametrů

Systém je navržen jako klient/server aplikace běžící na.NET Framework.

- Klientská část aplikace se spouští v prostředí prohlížeče HTML dokumentů (např. Internet Explorer, Firefox, a další. Obsahuje grafické rozhraní sloužící uživatelům prohlížet a spravovat uložená data (prezentační vrstva OSI). Aplikace je uložena na serveru a uživatelům je dostupná pomocí URL.
- Serverová část aplikace je navržena jako webová služba. Tvoří převážnou část aplikační logiky systému (aplikační vrstva OSI), správu dat (datová vrstva OSI). Webová služba je uložena na serveru a klientům je dostupná pomocí URL.

Z pohledu uživatele je aplikace rozdělena na dvě základní funkce:

a) Nahrávání datových souborů vzorku do informačního systému pomocí uživatelského rozhraní

b) Vizualizace vybraného vzorku a zobrazení statistických parametrů

- Zobrazení digitalizovaného snímku povrchu
- 3D zobrazení povrchu
- Tabulka statistických parametrů

3.1 Vývojové nástroje

Pro vývoj našeho informačního systému použijeme následující vývojové nástroje, které je potřeba rovněž nainstalovat:

- Visual Studio 2008 + SP1 (programování, ladění, kompilace)
- Silverlight3 tools (rozšíření VS2008 o podporu Silverlight)
- Expression Blend 3 (nástroj pro návrh grafického uživatelského rozhraní)

3.2 Server

Serverová aplikace se skládá z aplikační a komunikační vrstvy – webová služba (WCF) – a datové vrstvy – část webové služby a samotná databáze. Webová služba zajišťuje správu datových souborů, jež jsou uloženy přímo na disku serveru a jejichž hlavní atributy jsou zapsány

do databáze pro rychlejší vyhledávání. Webová služba je spuštěna pod webovým serverem aplikace v IIS.

3.2.1 Veřejné rozhraní webové služby – IWebMechService

Veřejné rozhraní webové služby je nejdůležitější část serverové aplikace a vlastně celého informačního systému, je jeho mozkem a srdcem. Přes veřejné rozhraní webové služby klienti komunikují se serverem, posílají dotazy a získávají odpovědi, data, informace. Proto se s touto části kódu seznámíme detailněji.

Rozhraní bychom mohli rozdělit na dvě části, metody pro správu povrchů a metody pro správu měřených vzorků. Takto jsou ostatně rozdělena i ve zdrojovém kódu.

Správa povrchů – IWebMechSurface

```
SurfaceData AddSurface(string name);
```

Uloží záznam o povrchu do DB. V případě úspěchu vrátí instanci třídy, odpovídající vloženým položkám záznamu v tabulce (obsahující i automaticky generované položky). Pokud dojde k chybě, vygeneruje patřičnou SoapException s chybovou hláškou.

Parametry:

`name` – jedinečné jméno povrchu

Návratová hodnota:

`SurfaceData` - objekt vloženého záznamu do tabulky Surface

```
void ModifySurface(SurfaceData surfaceData);
```

Opraví záznam o povrchu v DB. Pokud dojde k chybě, vygeneruje patřičnou SoapException s chybovou hláškou.

Parametry:

`surfaceData` – objekt změněného záznamu tabulky

```
void DeleteSurface(SurfaceData surfaceData);
```

Smaže záznam o povrchu z DB. Pokud dojde k chybě, vygeneruje patřičnou SoapException s chybovou hláškou.

Parametry:

`surfaceData` – objekt mazaného záznamu tabulky

```
SurfaceData GetSurfaceById(int surfaceId);
```

Vyhledá záznam o povrchu v DB. V případě úspěchu vrátí instanci třídy, odpovídající hledanému záznamu v tabulce. Pokud dojde k chybě, vygeneruje patřičnou SoapException s chybovou hláškou.

Parametry:

`surfaceId` – identifikační číslo povrchu

Návratová hodnota:

`SurfaceData` - objekt hledaného záznamu z tabulky Surface, nebo `null`.

`SurfaceData GetSurfaceByName(string name);`

Vyhledá záznam o povrchu v DB. V případě úspěchu vrátí instanci třídy, odpovídající hledanému záznamu v tabulce. Pokud dojde k chybě, vygeneruje patřičnou SoapException s chybovou hláškou.

Parametry:

`name` – jedinečné jméno povrchu

Návratová hodnota:

`SurfaceData` - objekt hledaného záznamu z tabulky Surface, nebo `null`.

`PageInfoData GetSurfaceListPageInfo(SearchFilterData filterData);`

Vyhledá záznamy povrchů v DB. V případě úspěchu vrátí instanci třídy, s informacemi o hledaných záznamech, jako je počet vyhledaných záznamů, index počátečního záznamu, apod. Pokud dojde k chybě, vygeneruje patřičnou SoapException s chybovou hláškou.

Parametry:

`filterData` – filter, podle kterého jsou vyhledávány povrchy v tabulce Surface.

Návratová hodnota:

`PageInfoData` – informace o hledané stránce z tabulky Surface.

`List<SurfaceData> GetSurfaceList(SearchFilterData filterData);`

Vyhledá záznamy povrchů v DB. V případě úspěchu vrátí jejich list. Pokud dojde k chybě, vygeneruje patřičnou SoapException s chybovou hláškou.

Parametry:

`filterData` – filter, podle kterého jsou vyhledávány povrchy v tabulce Surface.

Návratová hodnota:

`List<SurfaceData>` – list hledaných záznamů z tabulky Surface.

Správa měřených vzorků – IWebMechSample

`SampleData UploadSample(SurfaceData surfaceData,string imageName, byte[] imageArray, string pointsName, byte[] pointsArray);`

Uloží soubor obrázku, soubor s 3D body a vypočte statistické parametry, následně uloží záznam o měřeném vzorku do DB. V případě úspěchu vrátí instanci třídy, odpovídající vloženým položkám záznamu v tabulce (obsahující i automaticky generované položky). Pokud dojde k chybě, vygeneruje patřičnou SoapException s chybovou hláškou.

Parametry:

`surfaceData` – objekt nadřazeného povrchu

`imageName` – jméno souboru obrázku

`imageArray` – bytové pole souboru obrázku

`pointsName` – jméno souboru 3D bodů

`pointsArray` – bytové pole souboru 3D bodů

Návratová hodnota:

`SampleData` - objekt vloženého záznamu do tabulky Sample

```
void ModifySample(SampleData sampleData, string imageName,
            byte[] imageArray, string pointsName, byte[] pointsArray);
```

Upraví soubor obrázku, soubor s 3D body a vypočte nové statistické parametry, následně opraví záznam o měřeném vzorku v DB. Pokud dojde k chybě, vygeneruje patřičnou SoapException s chybovou hláškou.

Parametry:

`surfaceData` – objekt nadřazeného povrchu

`imageName` – jméno souboru obrázku

`imageArray` – bytové pole souboru obrázku

`pointsName` – jméno souboru 3D bodů

`pointsArray` – bytové pole souboru 3D bodů

```
void DeleteSample(SampleData sampleData);
```

Smaže záznam o měřeném vzorku z DB, datové soubory ponechá na disku. Pokud dojde k chybě, vygeneruje patřičnou SoapException s chybovou hláškou.

Parametry:

`sampleData` - objekt mazaného záznamu z tabulky Sample

```
PageInfoData GetSampleListPageInfo(SearchFilterData filterData,
            SurfaceData surfaceData);
```

Vyhledá záznamy měřených vzorků v DB. V případě úspěchu vrátí instanci třídy, s informacemi o hledaných záznamech, jako je počet vyhledaných záznamů, index počátečního záznamu, apod. Pokud dojde k chybě, vygeneruje patřičnou SoapException s chybovou hláškou.

Parametry:

`filterData` – filter, podle kterého jsou vyhledávány vzorky v tabulce Sample.

`surfaceData` – objekt nadřazeného povrchu

Návratová hodnota:

`PageInfoData` – informace o hledané stránce z tabulky Sample.

```
List<SampleData> GetSampleList(SearchFilterData filterData,
            SurfaceData surfaceData);
```

Vyhledá záznamy měřených vzorků v DB. V případě úspěchu vrátí jejich list. Pokud dojde k chybě, vygeneruje patřičnou SoapException s chybovou hláškou.

Parametry:

`filterData` – filter, podle kterého jsou vyhledávány vzorky v tabulce Sample.

`surfaceData` – objekt nadřazeného povrchu

Návratová hodnota:

`List<SampleData>` – list hledaných záznamů z tabulky Sample.

```
List<PointData> GetSample(SampleData sampleData);
```

Vyhledá záznamy 3D bodů měřeného vzorku v DB. V případě úspěchu vrátí jejich list. Pokud dojde k chybě, vygeneruje patřičnou SoapException s chybovou hláškou.

Parametry:

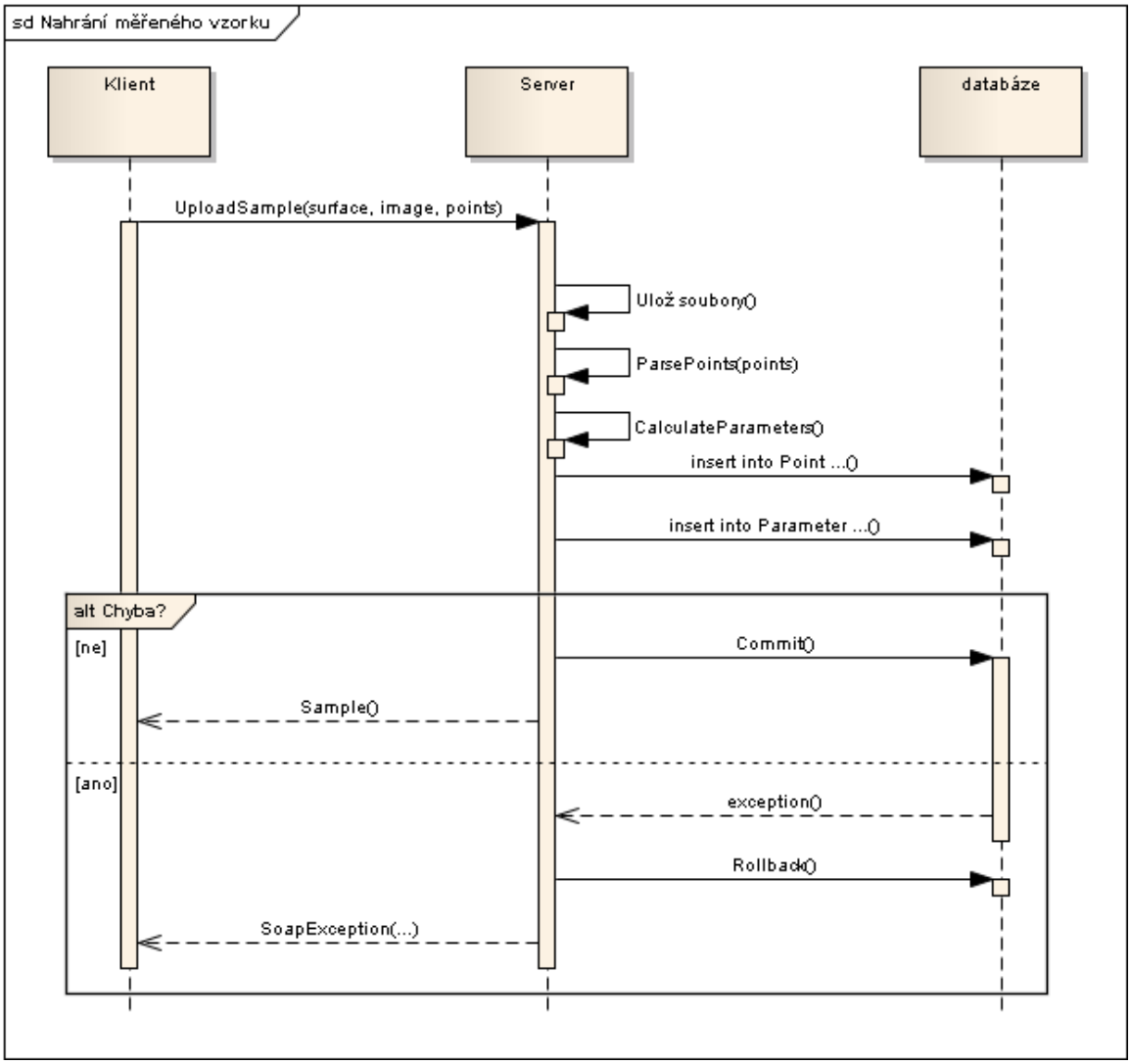
`sampleData` – objekt hledaného záznamu z tabulky Sample

Návratová hodnota:

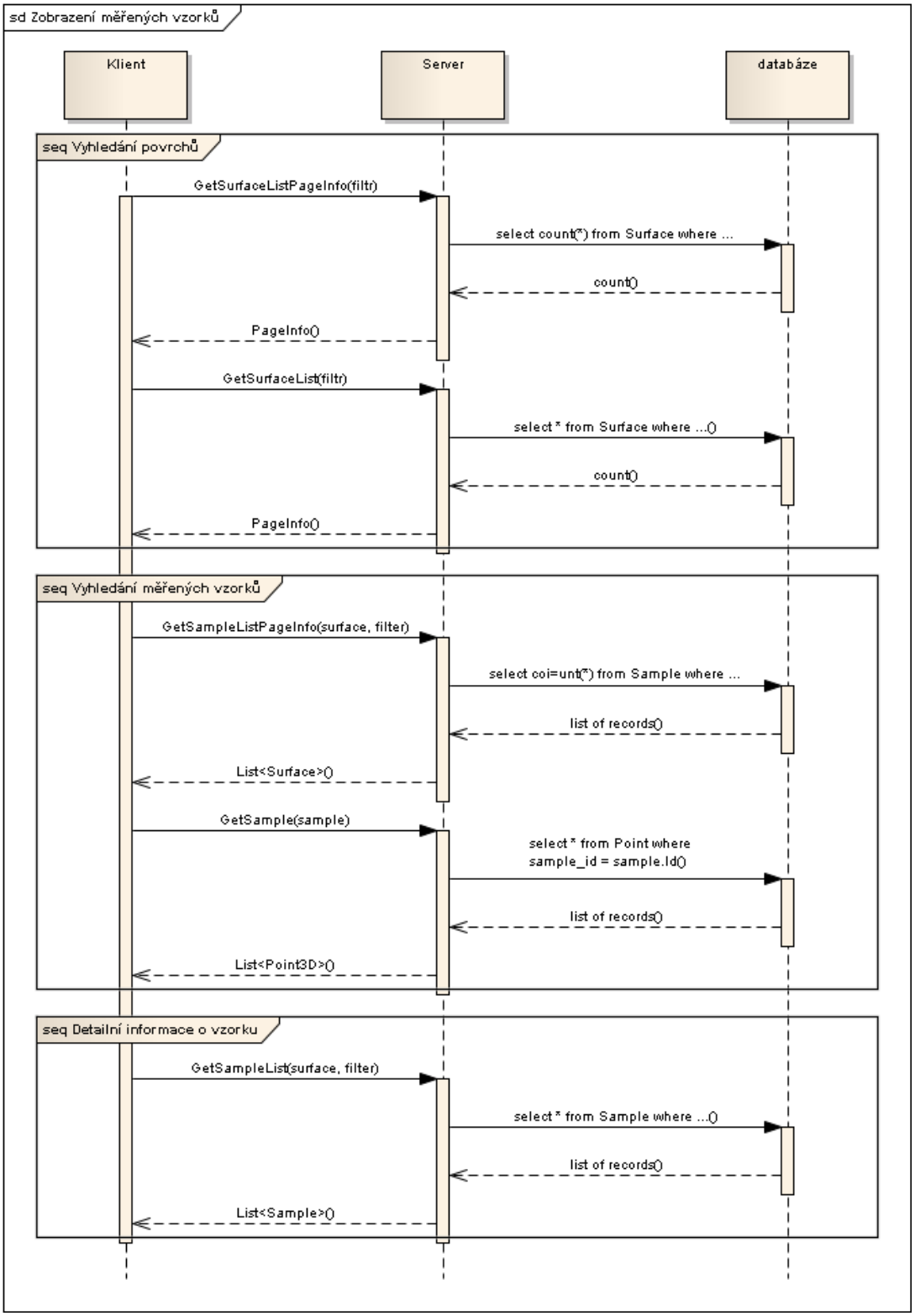
`List<PointData>` – list hledaných záznamů z tabulky Sample.

3.2.2 Sekvenční diagramy

Většina operací webové služby je triviální a není potřeba pro ně vytvářet názorné sekvenční diagramy. Pro lepší pochopení si uvedeme pouze dva, Zobrazení měřeného vzorku, kde popíšeme celý proces získání 3D povrchu, jeho dat a statistických parametrů, a také Nahrání měřeného vzorku do systému, abychom si představili, jak se vytvářejí jednotlivé data a soubory.



Obrázek 3-1: Sekvenční diagram - nahrání měřeného vzorku

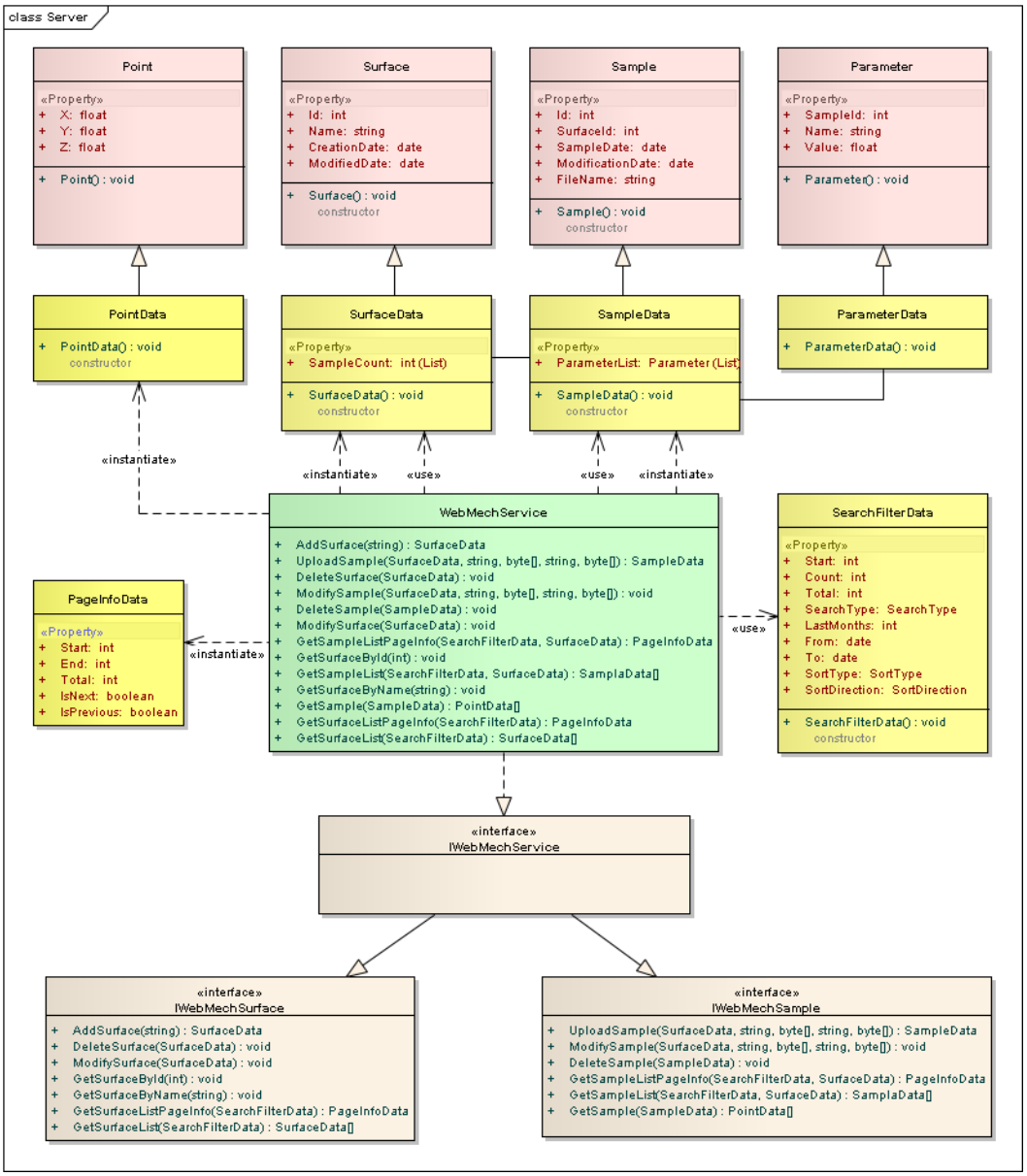


Obrázek 3-2: Sekvenční diagram - vyhledání měřených vzorků

3.2.3 Třídní diagram

Třídy webové služby (Obrázek 3-3) jsou rozděleny do tří skupin.

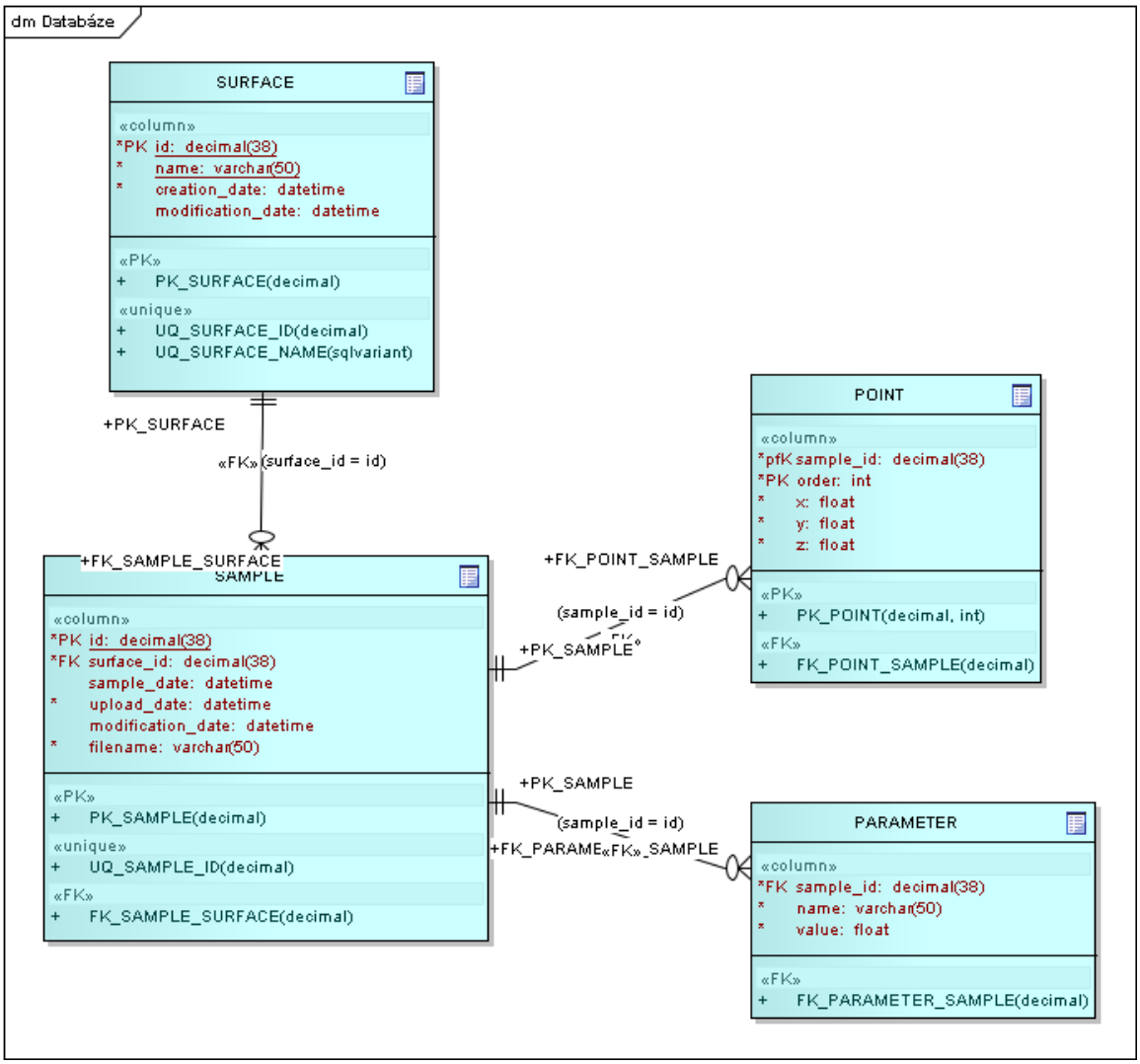
- Třídy databáze, kterou generoval průvodce LINQ to SQL, ty odpovídají definicím tabulek v databázi a pro ORM. V diagramu jsou zobrazeny růžovou barvou.
- Třídy datové, které slouží pro přenos dat mezi klientem a serverem. V diagramu jsou zobrazeny žlutou barvou.
- Třída komunikační, která slouží jako rozhraní mezi klientem a serverem, tato třída implementuje komunikační rozhraní složené ze dvou dílčích. V diagramu jsou zobrazeny zelenou a hnědou barvou.



Obrázek 3-3: Třídní diagram - server

3.2.4 Systém řízení báze dat

Databáze systému se skládá ze čtyř tabulek. Z konceptuálního schématu můžeme rozpoznat jejich strukturu a vazby. V tabulce Surface vedeme záznamy povrchů, jde o číselník, který nám hierarchicky organizuje měřené vzorky. Tabulka Sample obsahuje záznamy konkrétních měřených vzorků, obsahuje cizí klíč surface_id. Tabulka Parameter obsahuje záznamy vypočítaných statistických parametrů, obsahuje cizí klíč sample_id. A nakonec tabulka Point obsahuje záznamy nasnímaných 3D bodů, sloužící pro vizualizaci povrchu, obsahuje cizí klíč sample_id.



Obrázek 3-4: Databáze - konceptuální schéma

Tabulky rovněž popíšeme datovým modelem

SURFACE

atribut	datový typ	primární klíč	unikátní klíč	nepoviný atribut	cizí klíč	index	validace
id	integer	A	A	-	-	A	-
name	varchar	-	A	-	-	A	-
creation_date	datetime	-	-	-	-	-	-
modification_date	datetime	-	-	-	-	-	-

SAMPLE

atribut	datový typ	primární klíč	unikátní klíč	nepoviný atribut	cizí klíč	index	validace
id	integer	A	A	-	-	A	-
surface_id	integer	-	-	-	A	-	-
sample_date	datetime	-	-	-	-	-	-
upload_date	datetime	-	-	-	-	-	-
modification_date	datetime	-	-	-	-	-	-
filename	integer	-	-	-	-	-	-

PARAMETER

atribut	datový typ	primární klíč	unikátní klíč	nepoviný atribut	cizí klíč	index	validace
sample_id	integer	-	-	-	A	-	-
jmeno	varchar	-	-	-	-	-	-
ulice	float	-	-	-	-	-	-

POINT

atribut	datový typ	primární klíč	unikátní klíč	nepoviný atribut	cizí klíč	index	validace
sample_id	integer	A	-	-	A	-	-
order	integer	A	-	-	-	-	-
x	float	-	-	-	-	-	-
y	float	-	-	-	-	-	-
z	float	-	-	-	-	-	-

Tyto tabulky jsme implementovali do SQL Serveru 2005 Express Edition firmy Microsoft. Jelikož tabulky jsou poměrně jednoduché, nebylo potřeba vytvářet sql skripty. Vytvořili jsme je přímo v nástroji SQL Server Management Studio Express (není součástí instalace SQL Server 2005), rovněž od firmy Microsoft. Tento nástroj umožňuje mimo jiné zálohovat celou databázi, takže je vcelku jednoduché přenášet změny v tabulkách a uložená data z lokálního počítače na server a naopak.

Pro práci s databází nebyly definovány žádné složitější operace, nebyly analyzovány žádné transakce, všechny operace `select`, `insert`, `update`, `delete` jsou triviální. V našem řešení jsme použili objektově-relační mapování (ORM), které se .NET Frameworku nazývá LINQ to

SQL. Zde bych rád vyzdvihl jednoduchost skládání složitějších příkazů, nejlépe na fragmentu zdrojového kódu, pro výběr povrchu z databáze.

```
private IQueryable<Surface> GetSurfaceQuery(WebMechDataContext db,
                            SearchFilterData filter, bool allRows)
{
    // basic query from Surface table
    IQueryable<Surface> query = (from s in db.Surfaces select s);

    // filtering last X months
    if (filter.SearchType == SearchType.LastMonths)
        query = query.Where(s =>
            s.modification_date > DateTime.Now.Date.AddMonths
                                                  (-filter.LastMonths));
    // filtering - date range
    else if (filter.SearchType == SearchType.DateRange)
        query = query.Where(s =>
            (s.creation_date >= filter.From) &&
            (s.creation_date <= filter.To));

    // sorting by name
    if (filter.SortType == SortType.Name)
    {
        if (filter.SortDirection == SortDirection.Ascending)
            query = query.OrderBy(s => s.name);
        else
            query = query.OrderByDescending(s => s.name);
    }
    // sorting by date
    else if (filter.SortType == SortType.Date)
    {
        if (filter.SortDirection == SortDirection.Ascending)
            query = query.OrderBy(s => s.creation_date);
        else
            query = query.OrderByDescending(s => s.creation_date);
    }
    // sorting by number of samples
    else if (filter.SortType == SortType.NumberOfSamples)
    {
        if (filter.SortDirection == SortDirection.Ascending)
            query = query.OrderBy(s => s.Samples.Count);
        else
            query = query.OrderByDescending(s => s.Samples.Count);
    }

    // return all rows
    if (allRows)
        return query;
    // return range of rows e.g. 11-20
    return query.Skip(filter.Start - 1).Take(filter.Count);
}
```

Na začátku se vygenerujeme jednoduchý `select` všech řádků v tabulce Surface. Poté, pokud je tak specifikováno parametrem `filter`, se jeho výběr omezí pro požadované časové období

(několik měsíců nazpět, nebo v definovaném časovém horizontu). Následuje seřazení řádků, specifikováno parametrem `filter`, podle jména, data, či počtu uložených vzorků. Na závěr vrátíme dotaz pro všechny řádky, nebo dotaz omezený výběrem požadovaných řádků – tzv. stránkování, které se využívá v GUI, aby zobrazený seznam povrchů nebyl příliš dlouhý.

3.3 Klient

Klientská aplikace se skládá z prezentační vrstvy – uživatelské rozhraní aplikace, navržené pomocí WPF – a z komunikační vrstvy – klient webové služby. Ostatní vrstvy zde nejsou přítomny, jelikož hlavní aplikační logiku systému zajišťuje samotná webová služba.

3.3.1 Diagram užití

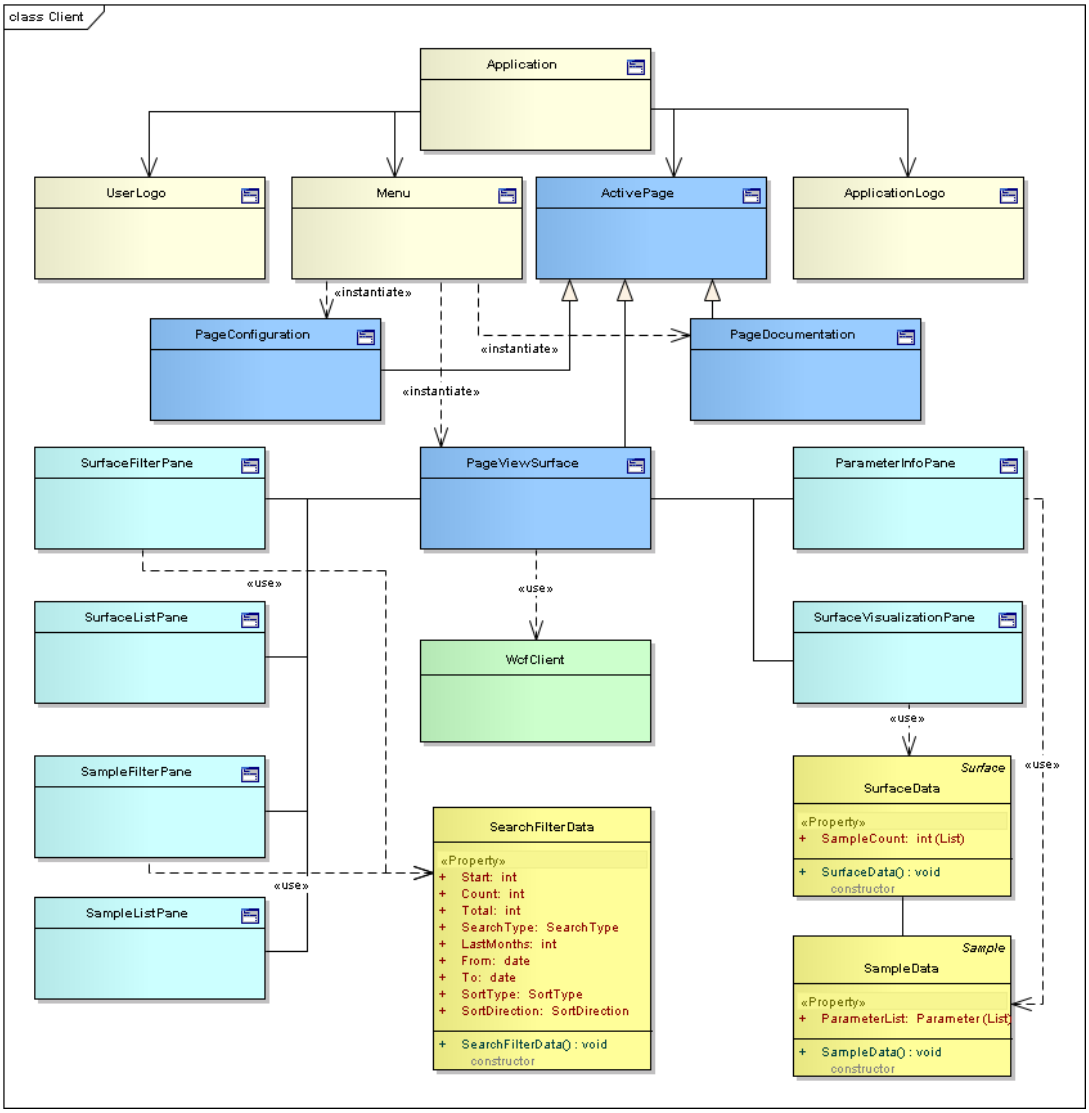
V původní analýze byly identifikovány dvě role (administrátor a uživatel), které si rozdělily několik procesů. V průběhu implementace, při upřesňování požadavků se zadavatelem vyplynulo, že role administrátor je v tuto chvíli zbytečná, a tak systém má definovanou pouze roli uživatel. Obrázek 3-5 zobrazuje všechny procesy systému a i přes jejich zdánlivě nízký počet, plní veškeré požadavky informačního systému.

Obrázek 3-5: Diagram případů užití

3.3.2 Třídní diagram

Třídy klientské aplikace (Obrázek 3-6) jsou rozděleny do tří hlavních skupin.

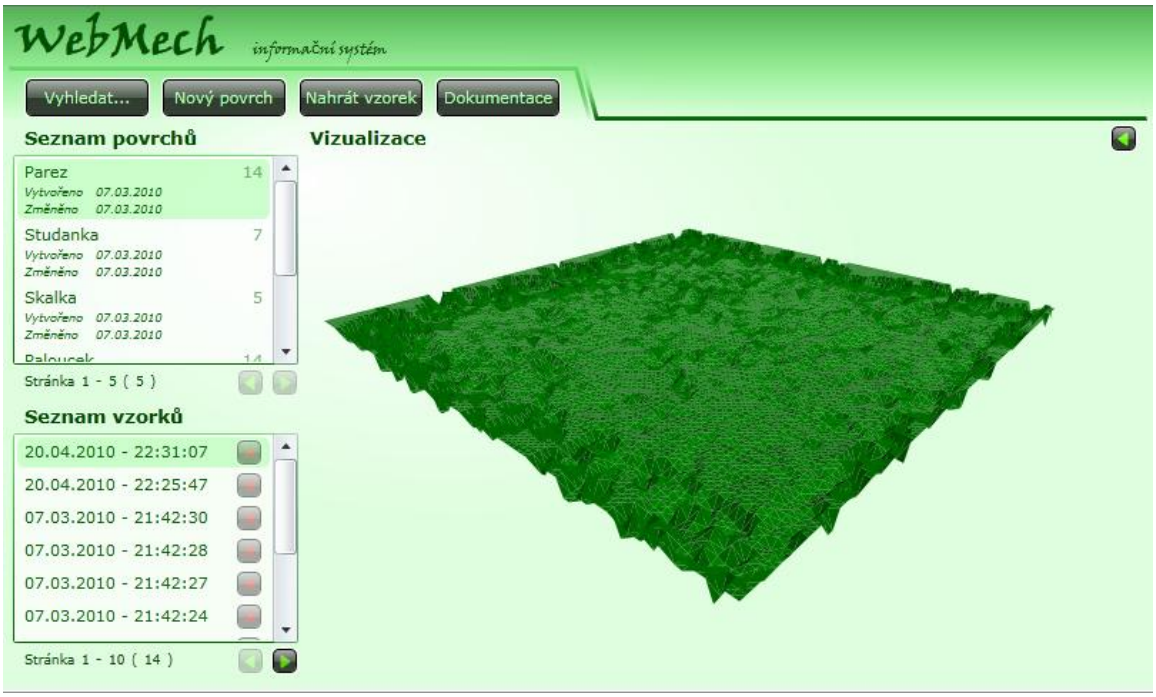
- Třídy grafického uživatelské rozhraní jsou rozděleny na třídy, které jsou společné pro celou aplikaci (bílou barvou), na stránky (modrou barvou) a detaily stránky zobrazující povrchy, zobrazeny zelenomodrou barvou.
- Třídy datové, které slouží pro přenos dat mezi klientem a serverem, jsou do klientské aplikace automaticky vloženy, přes WebResource pomocí WDSL a jsou shodné s datovými třídami ve webové službě. V diagramu jsou zobrazeny žlutou barvou.
- Třída komunikační, která slouží jako rozhraní mezi klientem a serverem, je do klientské aplikace automaticky vložena, přes WebResource pomocí WDSL. V diagramu je zobrazena zelenou barvou.



Obrázek 3-6: Třídní diagram - klient

3.3.3 Uživatelské rozhraní

Uživatelské rozhraní klientské aplikace je definováno pomocí WPF [10] a je složeno z několika částí. Třídou hlavního okna, která obsahuje grafické prvky názvu aplikace, tlačítek hlavního menu a prostoru pro právě zobrazenou stránku aplikace. Stránka pro výběr a vizualizaci povrchu je rozdělena na tři sloupce, vlevo jsou prvky pro výběr povrchu, uprostřed pak vizualizační objekt zobrazující buď obrázek, nebo 3D povrch a vpravo je tabulka statistických parametrů povrchu. Složením jednotlivých komponent získáme finální vzhled aplikace viz. Obrázek 3-7.



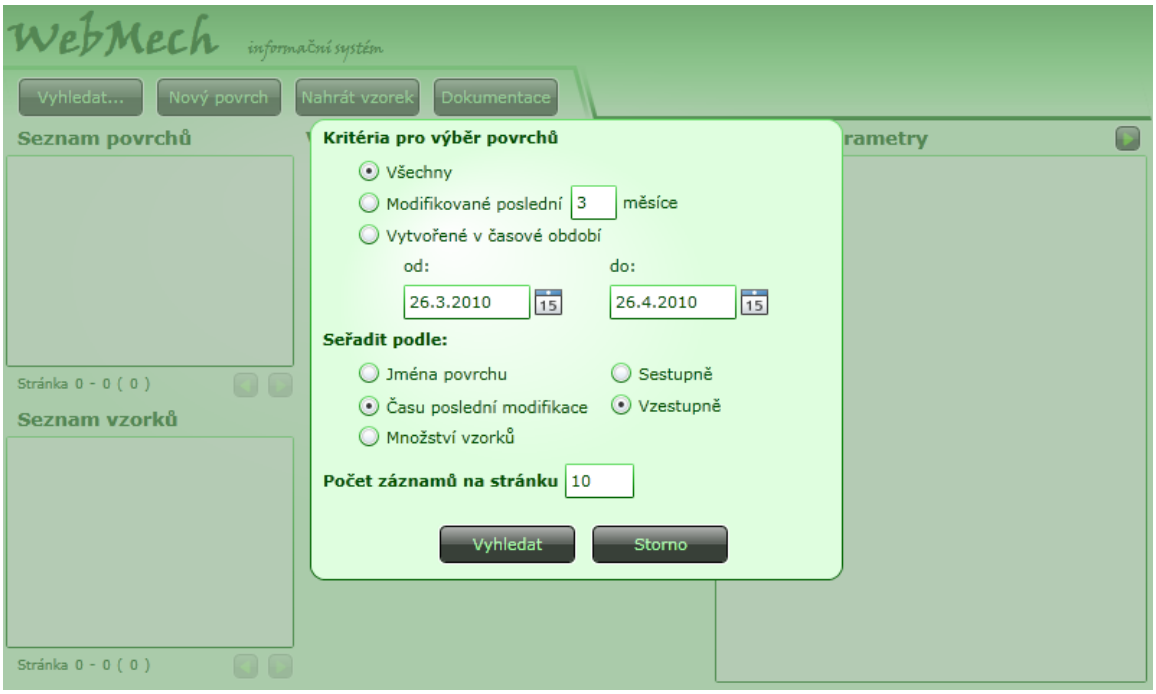
Obrázek 3-7: Uživatelské rozhraní klientské aplikace

Zajímavou – i když ne zcela novou – vlastností WPF je binding (vazba) mezi datovými třídami a uživatelským rozhraním [7]. Tuto vlastnost si představíme na příkladu. Jediná malá aplikační logika, kterou klient obstarává, je plnění předem vytvořených objektů – propojených s uživatelským rozhraním – daty, staženými z webové služby. Takže pokud uživatel stiskne tlačítko „Vyhledat“, klient zobrazí patřičný dialog (Obrázek 3-8). Objekt třídy `SearchFilterData` je rovněž uložen ve vlastnosti `DataContext` dialogu, která následně poskytuje data uložená v tomto objektu ostatním prvkům dialogu. Prvky pak zobrazují taková data, se kterými jsou svázána. V tomto případě je však binding definovaný jako obousměrný, což má za následek, že pokud uživatel změní hodnotu např. prvku „Datum od“, projeví se tato změna ihned i v objektu třídy `SearchFilterData`, bez nutnosti psaní dalšího kódu.

Prohlédněme si tedy fragment kódu ze souboru xaml.

```
<DatePicker
    ..
    SelectedDate="{Binding From, Mode=TwoWay,
                  UpdateSourceTrigger=Default}"
    ..
/>
```

Propojili jsme grafický uživatelský prvek `DatePicker` s položkou `From` datové struktury `SearchFilterData` (tento objekt je vložen do `DataContext` dialogu). Toto propojení jsme definovali jako obousměrné, přičemž modifikace dat probíhá defaultním způsobem.



Obrázek 3-8: Dialog "Vyhledat"

4 Instalace a konfigurace

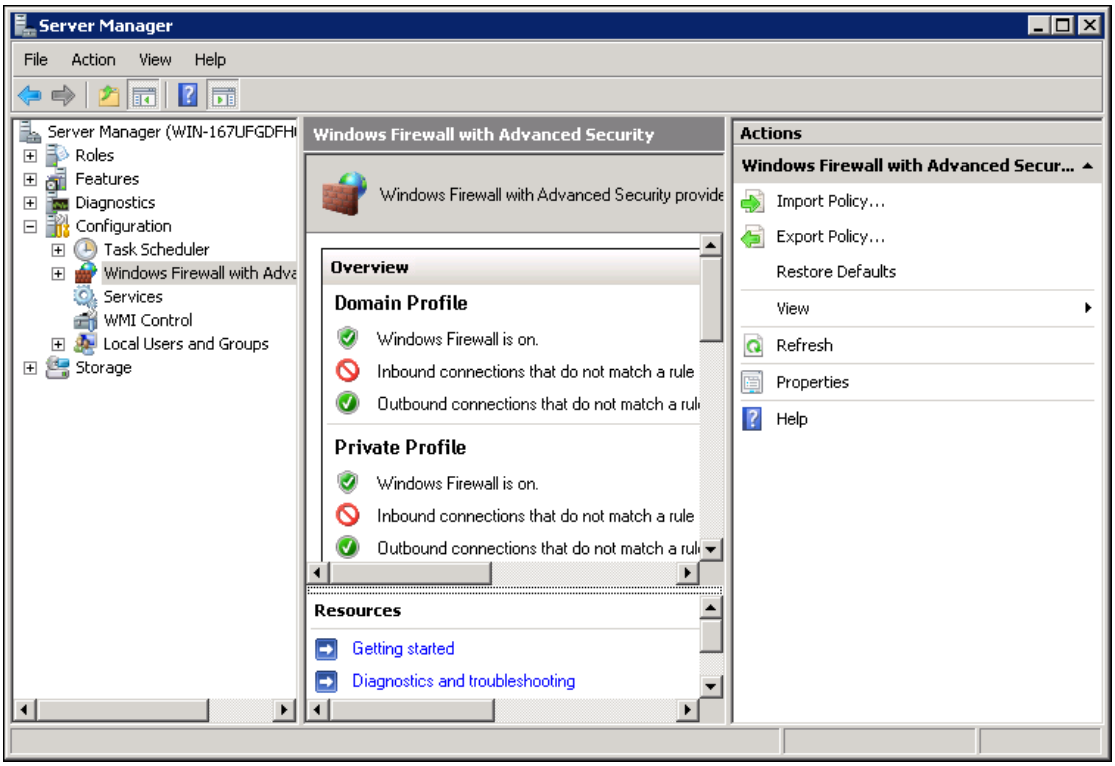
4.1 Instalace na straně serveru

Dříve, než instalujeme serverovou aplikaci, je nutné přichystat si počítač. Náš informační systém je vytvořen pro prostředí Windows firmy Microsoft a od toho se bude také odvíjet instalace software. Připravíme si následující programy, které – kromě operačního systému – jsou volně stažitelné z oficiálních stránek Microsoftu.

- operační systém Windows Server Enterprise
- .NET 3.5 Framework
- MS SQL Server 2005 Express Edition – databáze
- MS SQL Server 2005 Management Studio Express Edition

Serverovou aplikaci budeme instalovat (kopírovat) ve chvíli, kdy nakonfigurujeme IIS pro náš webový server.

Po instalaci operačního systému překontrolujeme a případně upravíme všechna nastavení serveru. Pozor bychom si měli především dát na nastavení firewallu, aby nám některá jeho část neblokovala komunikaci, kterou naše aplikace používá. Jelikož naše aplikace používá pouze základní HTTP přenos, postačuje pokut tento port (80) povolíme také ve firewallu.



Obrázek 4-1: Instalace - firewall

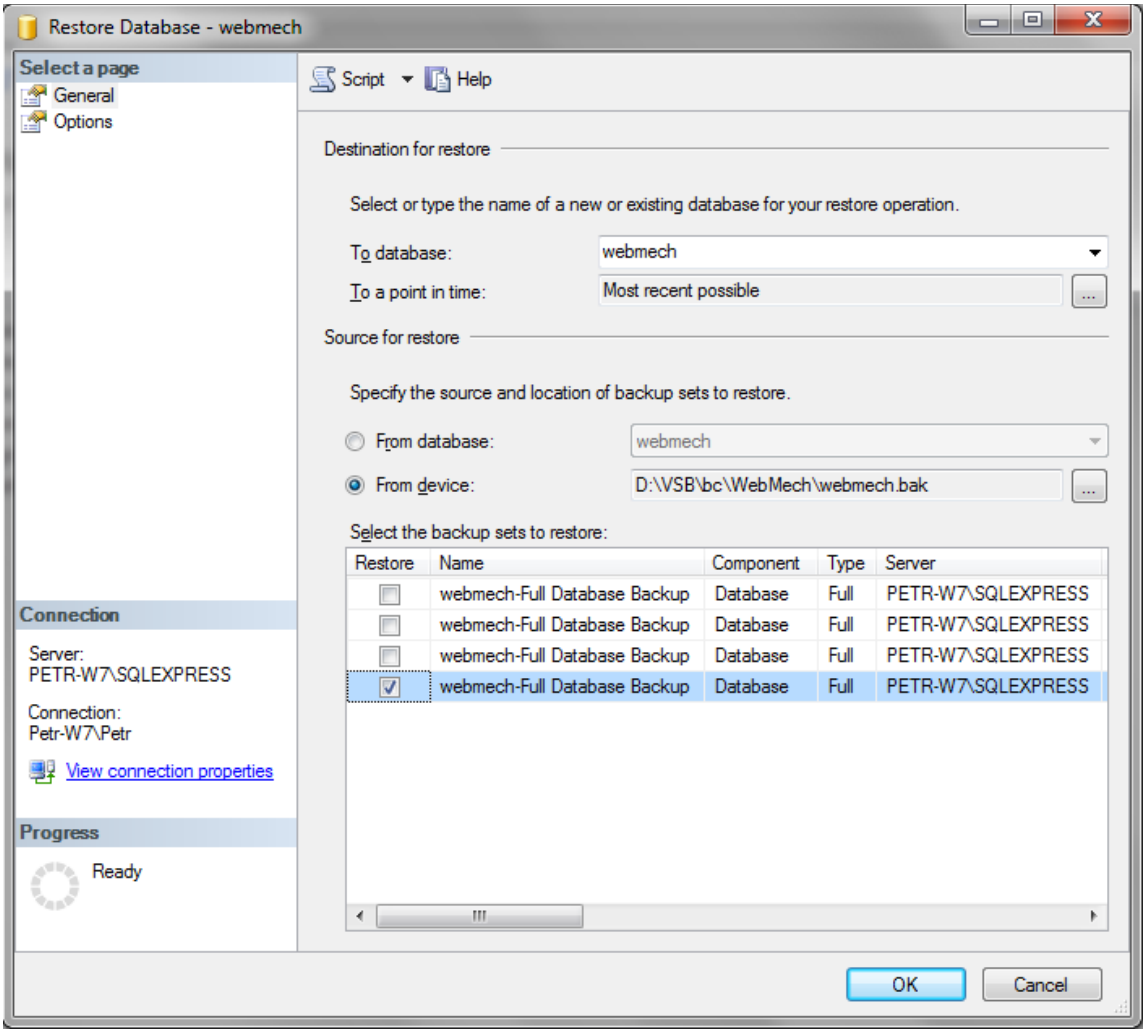
4.2 Konfigurace serverové části aplikace

V této části si ukážeme, jak konfigurovat základní stavební kameny aplikace.

4.2.1 MS SQL server

Pro základní práci naší aplikace nepotřebujeme nijak konfigurovat SQL server. Pokud by to však z jiných důvodů (např. bezpečnostních) bylo nutné, spustíme SQL Server Management Studio a vytvoříme nového uživatele, přiřadíme mu požadovaná práva a tabulky jednoduše převedeme pod jeho účet. Pozn. přihlašovací údaje nejsou součástí konfigurace a je potřeba znovu kompilovat projekt.

SQL Server Management Studio umí rovněž databáze zálohovat. Toho využijeme pro vytvoření a inicializaci naší databáze, kterou importujeme ze zálohy, obsahující prázdné tabulky.

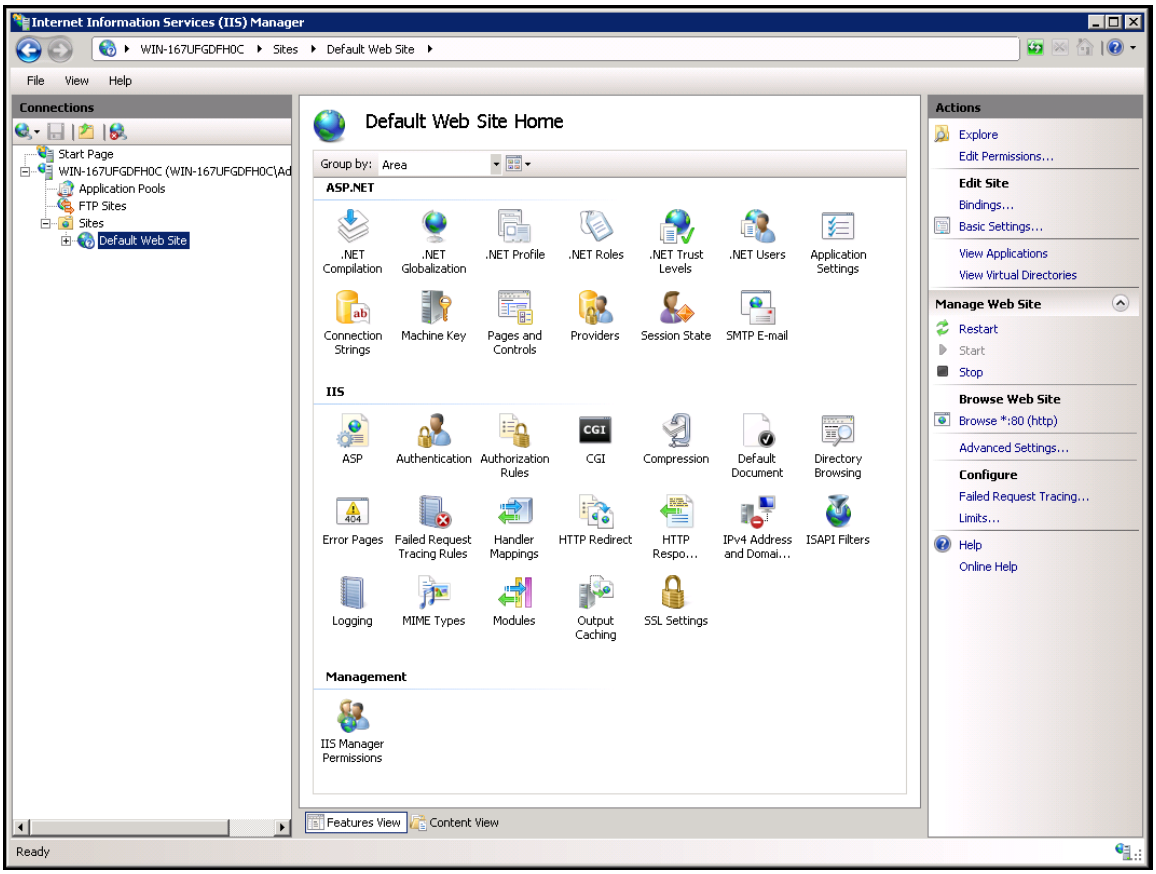


Obrázek 4-2: Obnovení databáze v SQL Serveru

4.2.2 IIS

Doinstalujeme IIS, pokud tomu tak není po instalaci operačního systému. Následující popis se vztahuje k anglické verzi Windows Server Enterprise. Spustíme program Server Manager, v levém stromu zvolíme položku Features a následně akci Add Features. Objeví se dialog, ve kterém zaškrtneme položku Remote Server Administration Tools → Role Administration Tools → Web Server (IIS) Tools a zvolenou funkci nainstalujeme.

Konfigurace samotného IIS je poněkud komplikovanější proces, který by vydal na zvláštní publikaci, proto si zde popíšeme pouze základ, který dostačuje ke zprovoznění naší serverové aplikace jako defaultního webového serveru. Obrázek 4-3 ukazuje konfigurační možnosti pro námi zvolený webový server.



Obrázek 4-3: Konfigurační menu IIS

První zkontrolujeme nastavení defaultních dokumentů, že se zde nachází default.aspx, který je úvodní stránkou našeho informačního systému a který nahrává Silverlight klienta do prohlížeče. Ostatní položky by již měly vyhovovat v defaultním nastavení.

4.2.3 Webový server a webová služba (Web.config)

Naši serverovou aplikaci – všechny její soubory – nakopírujeme do adresáře zvoleného webového serveru IIS. Pokud byla zachována defaultní instalace a zvolili jsme defaultní webový server, adresář by se měl nacházet na C:\inetpub\wwwroot.

Uvedeme si několik fragmentů konfiguračního souboru Web.config, které jsou pro správný běh aplikace nezbytné. Začněme s konfigurací adresářů, do kterých se nám budou ukládat soubory nasnímaných povrchů a jejich digitalizovaných 3D bodů.

```
<appSettings>
  <add key="samplesDir" value="files\samples" />
  <add key="imagesDir" value="files\images" />
</appSettings>
```

Tyto adresáře jsou relativní instalovanému webovému serveru. Ten by se měl nacházet, při zachování defaultní instalace, v adresáři C:\inetpub\wwwroot. Uděláme dobře, pokud vytvoříme adresář files ručně a přiřadíme mu práva pro čtení i zápis (uživatel IIS_IUSRS), aby naše webová služba neměla problémy při ukládání souborů.

Další důležitou položkou je připojovací řetězec k databázi. V této položce nezapomeneme změnit jméno našeho serveru za řetězec SERVER_NAME.

```
<add name="webmechConnectionString"
     connectionString="Data Source=SERVER_NAME\SQLEXPRESS;Initial
                       Catalog=webmech;Integrated Security=True"
     providerName="System.Data.SqlClient" />
```

Důležité je rovněž správné nastavení runtime modulu.

```
<httpRuntime maxRequestLength="65536"
             useFullyQualifiedRedirectUrl="true"
             executionTimeout="45" />
```

Poslední položkou, kterou musíme správně nakonfigurovat, je webová služba [9]. Pro basicHttpBinding správně nakonfigurujeme časové limity a velikosti bufferů. Toto je důležité, jelikož pomocí klientů přenášíme veliké soubory, desítky MB, a mohlo by docházet k přetečení těchto bufferů, nebo k překročení časů pro zpracování požadavku. Na druhou stranu musíme dát pozor, aby tyto hodnoty nebyly příliš vysoké a nezatěžovaly nepřiměřeně server.

```
<basicHttpBinding>
   <binding name="WebMechBindingBasic"
            closeTimeout="00:30:00"
            openTimeout="00:30:00"
            receiveTimeout="00:30:00"
            sendTimeout="00:30:00"
            maxReceivedMessageSize="33554432"
            maxBufferSize="33554432"
            maxBufferPoolSize="33554432"
            transferMode="Buffered" >
     <readerQuotas maxArrayLength="33554432"
```

```
                    maxBytesPerRead="33554432"
                    maxDepth="33554432"
                    maxNameTableCharCount="33554432"
                    maxStringContentLength="33554432"/>
      <security mode="None" />
    </binding>
  </basicHttpBinding>
```

Konfigurace samotné webové služby je již triviální

```
<service behaviorConfiguration="WebMech.Web.WebMechServiceBehavior"
         name="WebMech.Web.WebMechService">
  <endpoint address=""
            binding="basicHttpBinding"
            bindingConfiguration="WebMechBindingBasic"
            contract="WebMech.Web.IWebMechService">
  </endpoint>
</service>
```

4.3 Instalace na straně klienta

- operační systém Windows XP, Vista, 7, MacOS (pro operační systém linux sice existuje projekt Moonlight, jelikož však není 100% kompatibilní se Silverlight, k provozu aplikace se nedoporučuje)
- prohlížeč Internet Explorer, případně Firefox, Opera
- instalovaný Silverlight3 - zásuvný modul v prohlížeči

Klientská aplikace bude přístupna přes URL a bude se spouštět v prostředí prohlížeče HTML dokumentů. Instalace se v zásadě skládá ze dvou částí:

a) Pokud daný počítač nebude mít nainstalovaný zásuvný modul (Framework) Silverlight 3 bude instalační program automaticky stažen ze serveru (na kterém je nainstalovaná serverová aplikace a uživatel bude automaticky vyzván k jeho instalaci.

Pozn. Pokud není instalace Silverlight 3 úspěšně dokončena (chyba instalace, uživatel nemá oprávnění k instalování programů,…) nelze klientskou aplikaci na daném počítači spustit a je nutné kontaktovat správce sítě, počítačů.

b) Pokud je Framework Silverlight 3 úspěšně nainstalován, klientská aplikace se automaticky stáhne ze serveru a spustí v prohlížeči.

4.4 Konfigurace klientské části aplikace

Klientská část není tolik složitá na konfiguraci. Otevřeme adresář našeho webového serveru a zvolíme podadresář ClientBin, ve kterém se nachází xap soubor naší Silverlight aplikace, který se přenáší a spouští v klientském prohlížeči. V podstatě tento soubor není nic jiného, než přejmenovaný komprimační zip soubor. Pokud jej přejmenujeme na zip a otevřeme, jeden ze souborů, s názvem ServiceReferences.ClientConfig, je náš konfigurační. Ten otevřeme a editujeme. Po skončení práce nezapomeneme opět soubor přejmenovat na xap.

Samotný konfigurační soubor obsahuje pouze konfiguraci webové služby [6], ke které se klient připojuje, je krátký a tak jej zde uvedeme celý. Důležité je definovat správnou URL adresu webové služby, ostatní položky by se měly shodovat se stejnými položkami definovanými na straně serveru.

```
<configuration>
  <system.serviceModel>
    <bindings>
      <basicHttpBinding>
        <binding name="BasicHttpBinding_IWebMechService"
                 closeTimeout="00:30:00"
                 openTimeout="00:30:00"
                 receiveTimeout="00:30:00"
                 sendTimeout="00:30:00"
                 maxBufferSize="33554432"
                 maxReceivedMessageSize="33554432">
          <security mode="None">
            <transport>
              <extendedProtectionPolicy policyEnforcement="Never" />
            </transport>
          </security>
        </binding>
      </basicHttpBinding>
    </bindings>
    <client>
      <endpoint address="http://URL/WcfService/WebMechService.svc"
                binding="basicHttpBinding"
                bindingConfiguration="BasicHttpBinding_IWebMechService"
                contract="WebMechService.IWebMechService"
                name="BasicHttpBinding_Localhost" />
    </client>
  </system.serviceModel>
</configuration>
```

5 Závěr

Na tomto vcelku jednoduchém projektu jsme se setkali s několika technologiemi. Vytvořili jsme informační systém jako klient/server internetovou aplikaci. Tímto jsme oddělili aplikační vrstvu od prezentační a zároveň umožnili uživatelům využívat náš systém na libovolném počítači, ze kterého je možné připojit se k serveru pomocí URL. Klienta není třeba instalovat, je rovněž dostupný pomocí URL, ze kterého se stáhne přímo do internetového prohlížeče. To zda je server dostupný pouze pro vnitropodnikovou síť, nebo v celém světě pomocí internetu, již ponecháme na rozhodnutí při nasazování aplikace.

Informační systém je schopen evidovat naměřené vzorky povrchů a hierarchicky je uchovávat v databázi a soubory většího rozsahu na konfigurovaném paměťovém médiu. Klient má možnost ty vzorky spravovat, tzn. Ukládat vzorky nové, mazat staré, nepotřebné a hlavně prohlížet srovnávat vzorky uložené. Systém ke každému vzorku povrchu spočítá statistické parametry, které jsou zobrazovány klientem společně s nasnímaným obrázkem povrchu, či jeho 3D reprezentací.

Serverovou část aplikace tvoří hlavně webová služba, která je vytvořena na základech WCF, jež je součástí .NET Framework 3.5. Přenosové vlastnosti, jako i komunikační protokol je možné měnit pouze pomocí konfigurace, takže není potřeba nové kompilace projektu. Webová služba ukládá spravovaná data do databáze. Přistupuje k nim pomocí objektově-relačního mapování použitím knihovních tříd LINQ to SQL, které jsou rovněž součástí .NET Framework 3.5.

Klientská část aplikace je implementovaná jako internetová aplikace Silverlight. Tento Framework není přímo součástí .NET Frameworku, avšak tvoří jej některé třídy .NET Frameworku, které jsou speciálně kompilovány pro internetovou technologii. Grafické rozhraní je vytvořeno pomocí WPF, které umožňuje grafické prvky umístit na stránku a definovat jejich vlastnosti a chování. V neposlední řadě jsme implementovali klienta webové služby, jehož chování je možné ovlivňovat pomocí konfiguračního souboru.

Při zamyšlení nad možností rozšíření našeho informačního systému, mohli bychom doporučit následující funkce:

Automatický „uploader“ povrchů
Je systémová služba nainstalovaná na PC se sdíleným adresářem. Služba v plánovaném čase prohledává adresář, a pokud se v něm objeví nový datový soubor povrchu, pokusí se jej odeslat na server (využitím stejné webové služby používané klientskou internetovou aplikací pro nahrávání povrchů). Provedené operace se budou zaznamenávat do souboru „událostí“.

RSS (ATOM) čtečka
K webové službě serverové části systému může být připojena služba pro RSS (ATOM) čtečky, které lze následně umístit na libovolné stránky. Poskytované informace mohou být konfigurovatelné – poslední přidané povrchy, jejich statistické parametry, celková statistika, apod.

Literatura

[1] BĚHÁLEK, Marek. *Programovací jazyk C# - přednášky*. VŠB

[2] IC#Code. *SharpDevelop - The Open Source Development Environment for .NET*. http://www.icsharpcode.net/OpenSource/SD/Default.aspx.

[3] Lacko Ľuboslav. *Silverlight 3*. Czech MSDN. http://download.microsoft.com/download/C/0/6/C063ED26-AD71-4731-A6BA-791C6888BCD0/Silverligh_3.pdf.

[4] Michigan Metrology. *Confidential 3d Surface Roughness Measurement Analysis in One Week or Less.* http://www.michmet.com/S_Parameters.htm

[5] Microsoft. *MSDN*. http://msdn.microsoft.com.

[6] PAPA, John. *Silverlght: datové služby*. Zoner Press, 2009 s. ISBN 978-80-7413-041-0.

[7] PETZOLD Charles. *Mistrovství ve Windows Presentation Foundation*. Computer Press, 2009 s. ISBN 978-80-251-2141-2

[8] *.NET Framework Developer Center*. http://msdn.microsoft.com/en-us/netframework/default.aspx.

[9] *WCF – Windows Communication Foundation*. http://msdn.microsoft.com/en-us/netframework/aa663324.aspx.

[10] *WPF – Windows Presentation Foundation*. http://windowsclient.net/.